Pasteboards, Services, and Interoperability on Mac

Session 109

These are confidential sessions—please refrain from streaming, blogging, or taking pictures.

Pasteboards, Services, and Interoperability on Mac

James Dempsey
Cocoa Frameworks Engineer

Pasteboards and Services

- Pasteboards provide the foundation for data transfer between apps
- Services allow apps to extend each other's functionality
- Both have a long history in Mac OS X
- Both have significant enhancements in Snow Leopard

Only on Mac OS

What You'll Learn

- What's new with NSPasteboard and Services in Snow Leopard
- Using the new NSPasteboard APIs
- How to provide and consume Services in Snow Leopard

Only on Mac OS

Prerequisites

- Familiarity with
  - NSPasteboard
  - Uniform Type Identifiers (UTIs)
  - Property lists

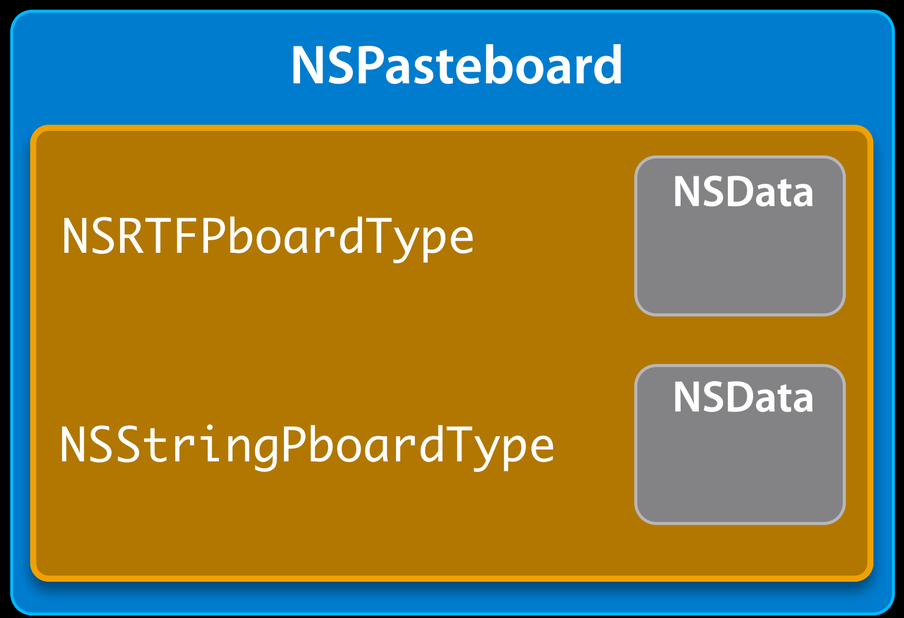
NSPasteboard

NSPasteboard

- Foundation for fundamental Mac features
  - Cut, copy, and paste
  - Drag and drop
  - System Services
- Allows for simple sharing of data between applications and within an application

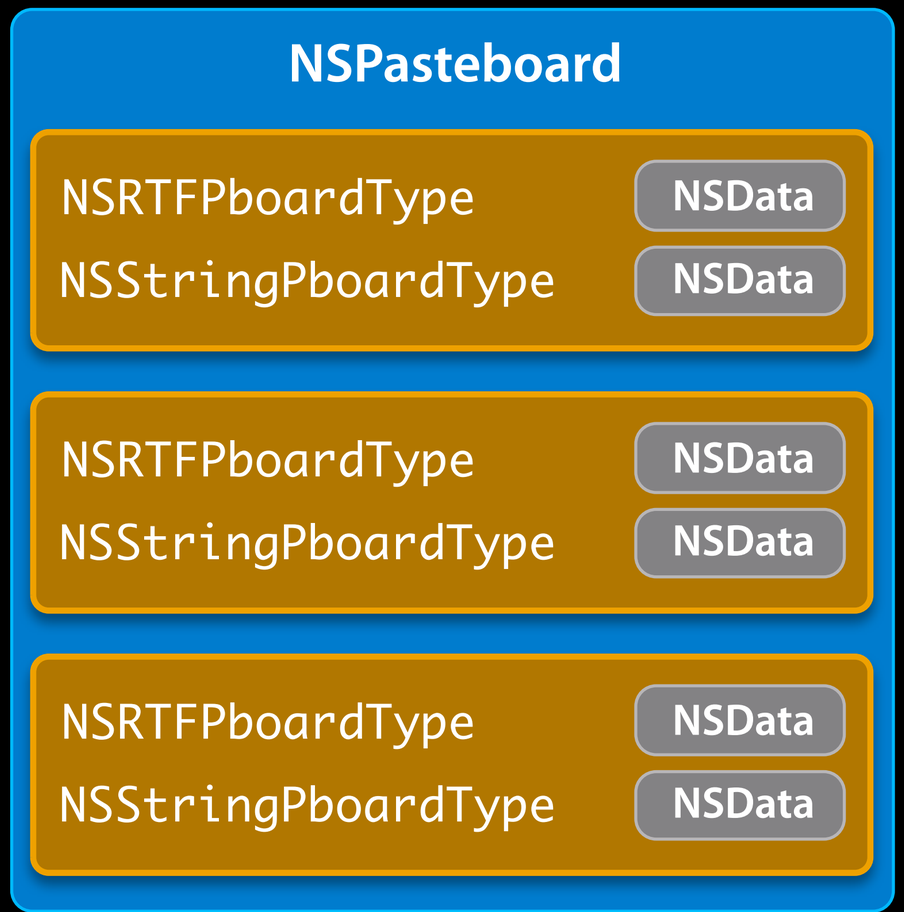
Only on Mac OS

NSPasteboard



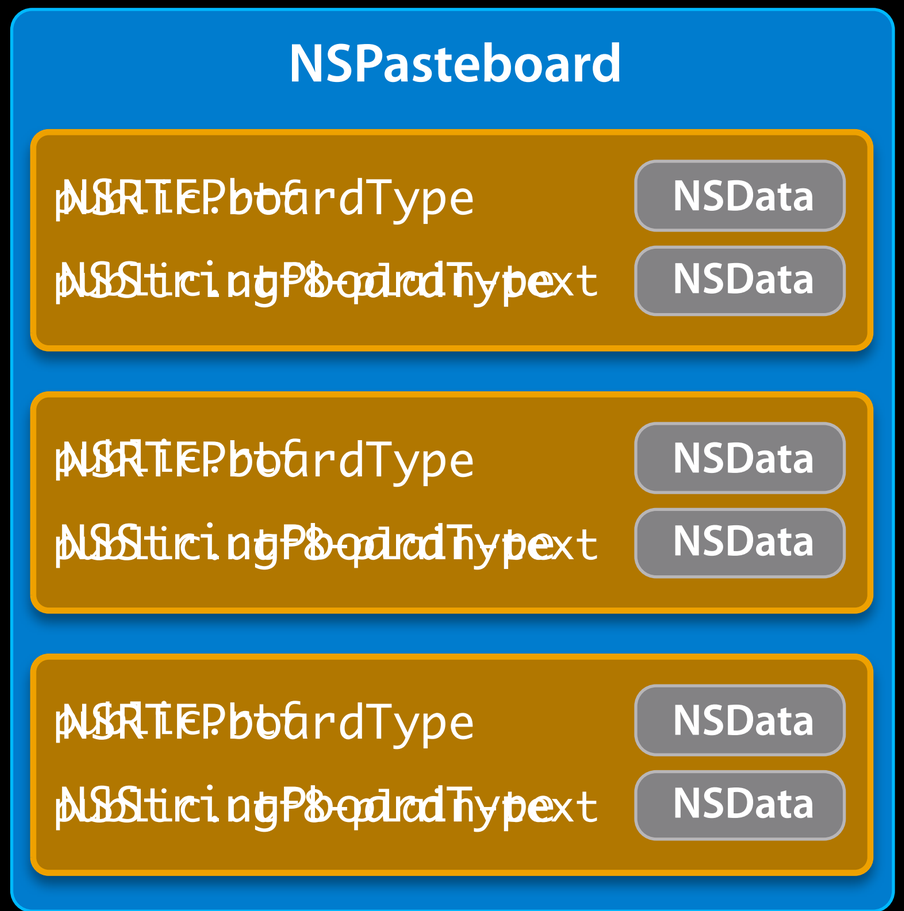
- Simple interprocess data exchange
- Single logical item
- Multiple data representations
- Data keyed by type
- Order of types is important

NSPasteboard in Snow Leopard



- Additions and changes
  - Multiple pasteboard items
  - Shift focus from types to classes
  - Move to Uniform Type Identifiers

NSPasteboard in Snow Leopard



- Additions and changes
  - Multiple pasteboard items
  - Shift focus from types to classes
  - Move to Uniform Type Identifiers
- Things to preserve
  - Ability to work with data types
  - Convenience conversions
  - Ability to provide data lazily

NSPasteboard

Writing to the pasteboard in Snow Leopard

NSURL

NSURL

NSURL

NSPasteboard

Writing to the pasteboard in Snow Leopard

NSURL

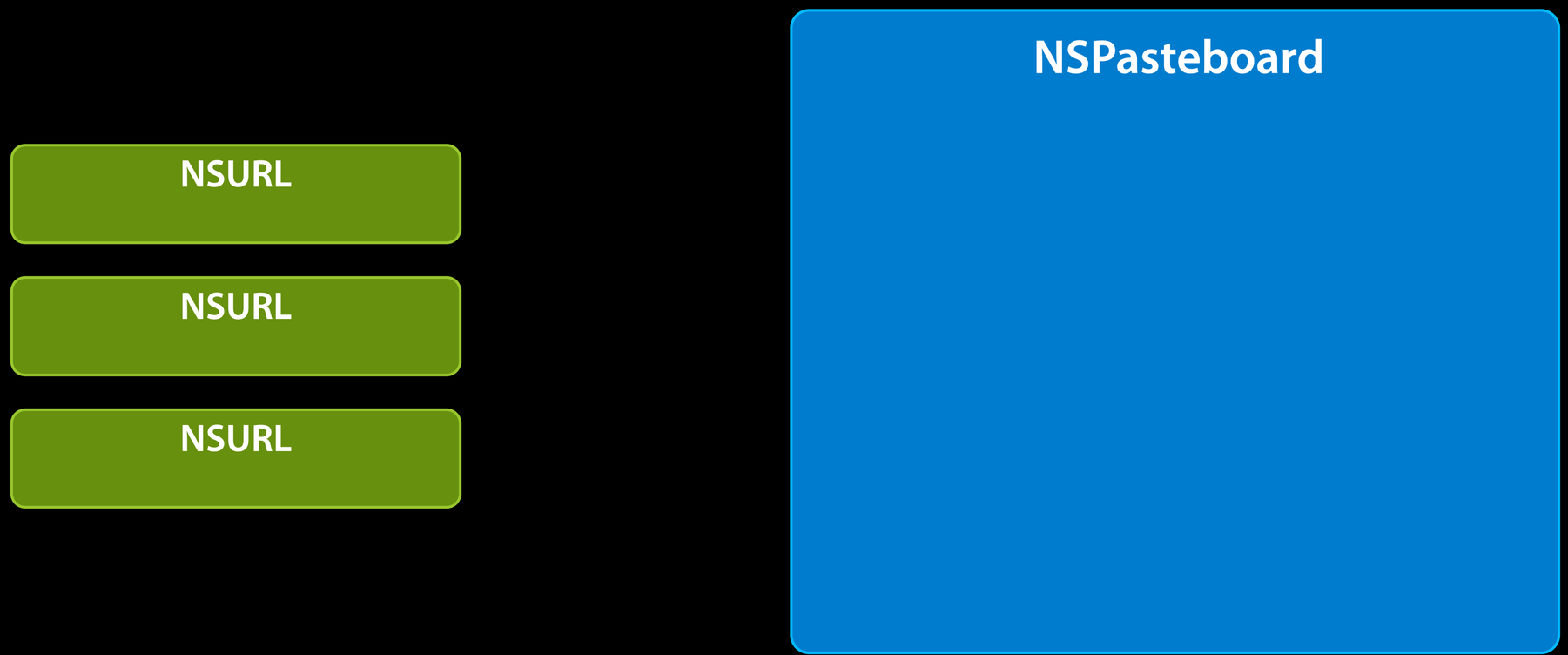
NSURL

NSURL

NSPasteboard

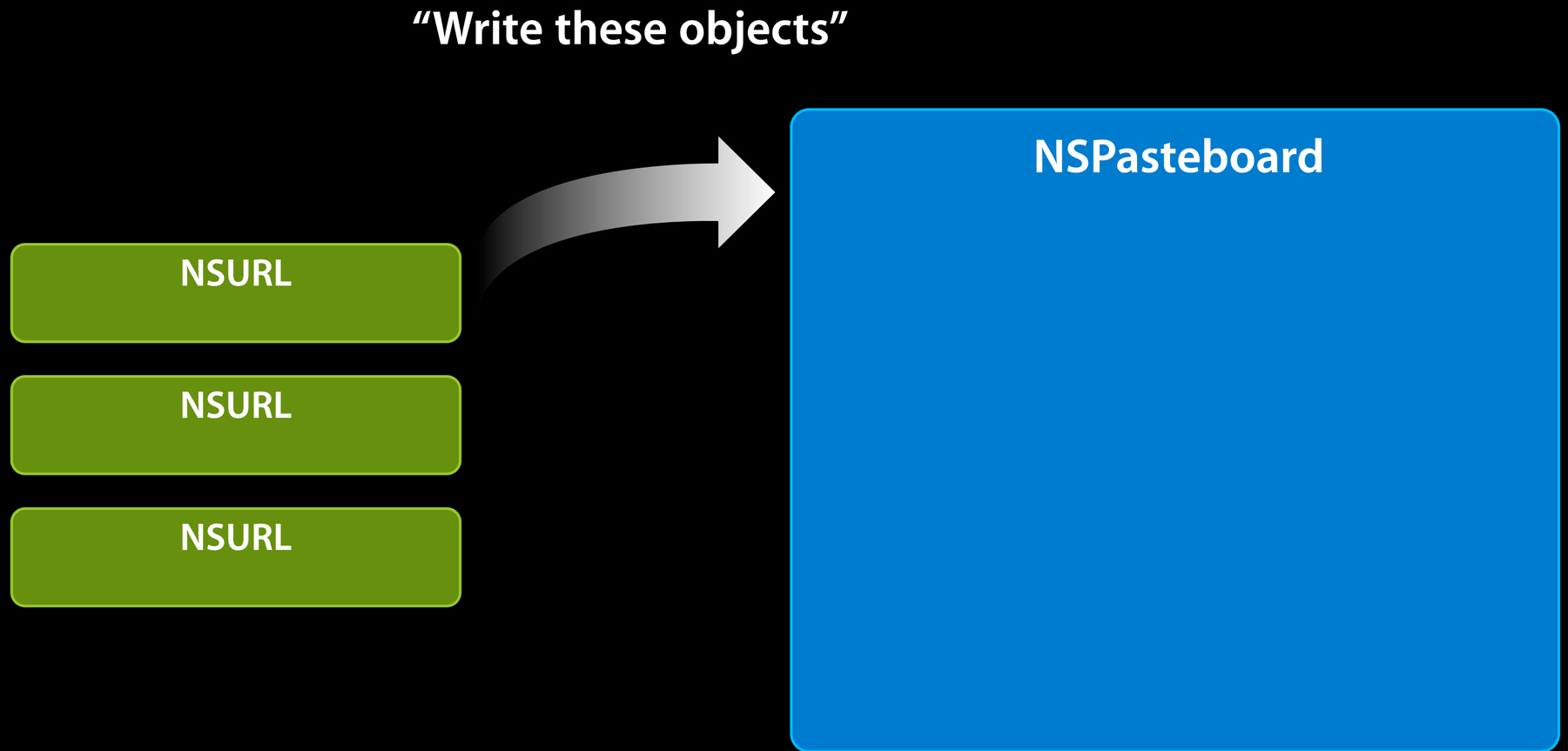
NSPasteboard

Writing to the pasteboard in Snow Leopard



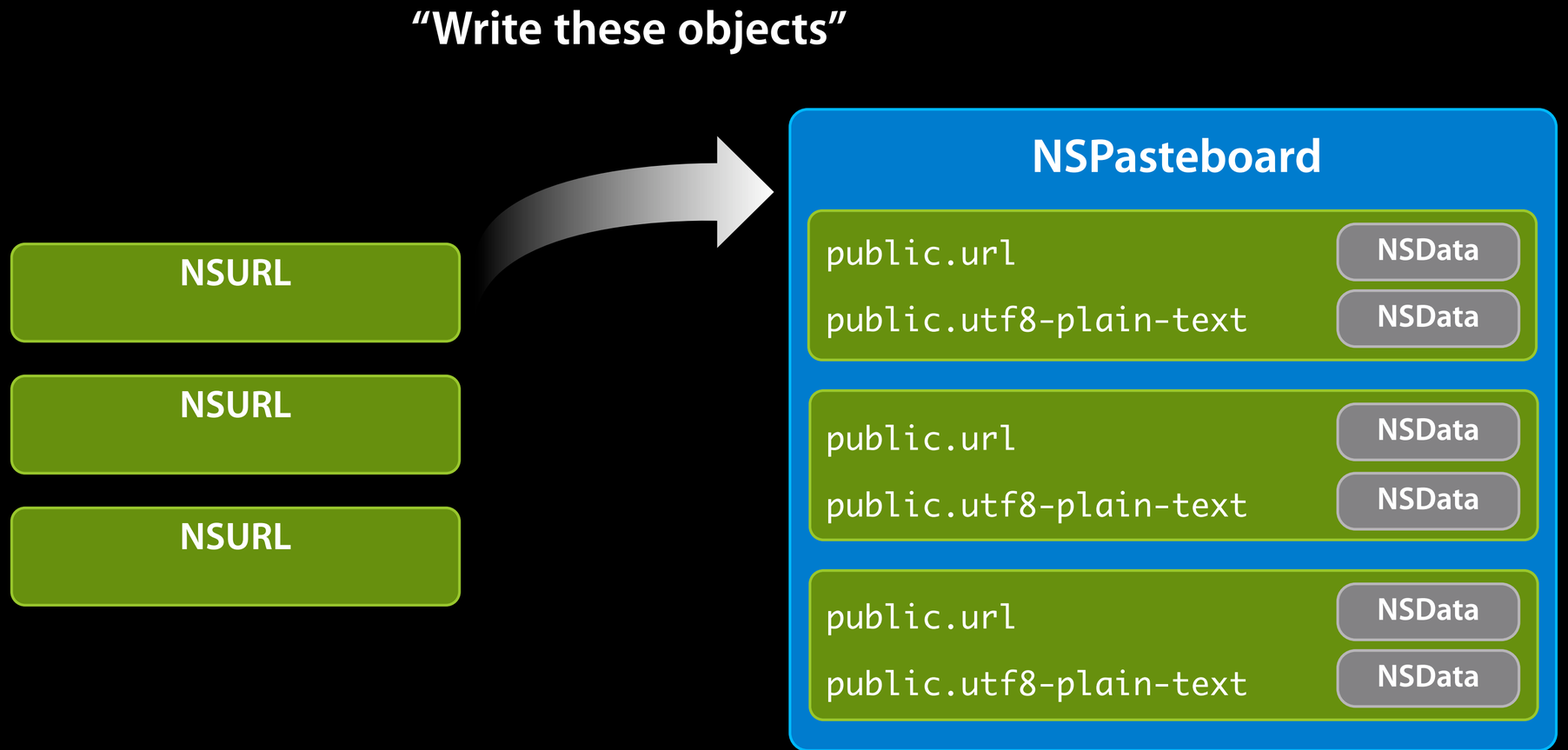
NSPasteboard

Writing to the pasteboard in Snow Leopard



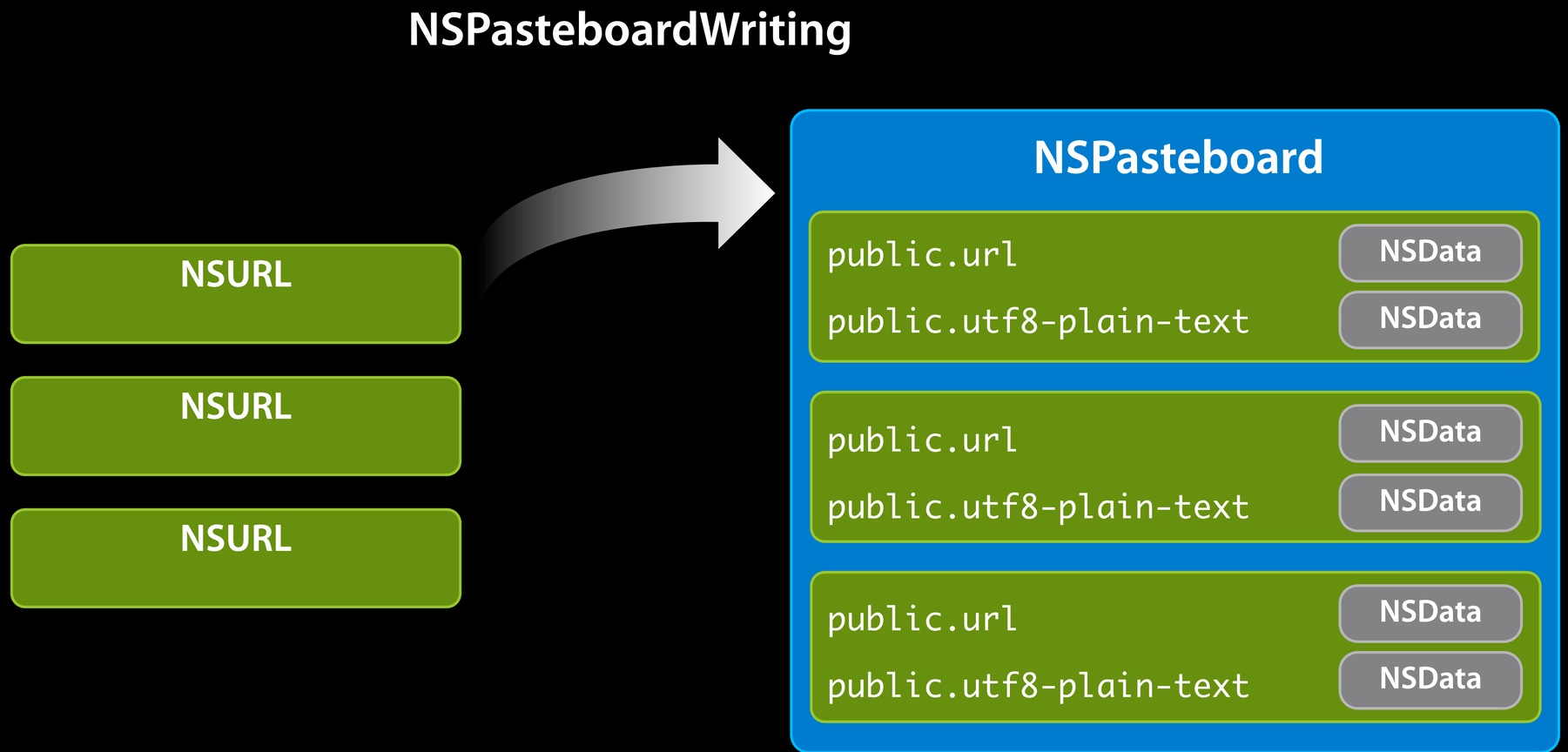
NSPasteboard

Writing to the pasteboard in Snow Leopard



NSPasteboard

Writing to the pasteboard in Snow Leopard



Snow Leopard Writing APIs

```
NSArray *URLs = ...;

NSPasteboard *pboard = [NSPasteboard generalPasteboard];

[pboard clearContents];

[pboard writeObjects:URLs];
```

Snow Leopard Writing APIs

```
NSArray *URLs = ...;

NSPasteboard *pboard = [NSPasteboard generalPasteboard];

[pboard clearContents];

[pboard writeObjects:URLs];
```

Snow Leopard Writing APIs

```
NSArray *URLs = ...;

NSPasteboard *pboard = [NSPasteboard generalPasteboard];

[pboard clearContents];

[pboard writeObjects:URLs];
```

Snow Leopard Writing APIs

```
NSArray *URLs = ...;

NSPasteboard *pboard = [NSPasteboard generalPasteboard];

[pboard clearContents];

[pboard writeObjects:URLs];
```

Snow Leopard Writing APIs

```
NSAttributedString *attString = ...;

NSPasteboard *pboard = [NSPasteboard generalPasteboard];

[pboard clearContents];

[pboard writeObjects:[NSArray arrayWithObject:attString]];
```

Snow Leopard Writing APIs

```
NSImage *image = ...;

NSPasteboard *pboard = [NSPasteboard generalPasteboard];

[pboard clearContents];

[pboard writeObjects:[NSArray arrayWithObject:image]];
```

Snow Leopard Writing APIs

```
NSColor *color = ...;

NSPasteboard *pboard = [NSPasteboard generalPasteboard];

[pboard clearContents];

[pboard writeObjects:[NSArray arrayWithObject:color]];
```

Snow Leopard Writing APIs

```
NSString *string = ...;

NSPasteboard *pboard = [NSPasteboard generalPasteboard];

[pboard clearContents];

[pboard writeObjects:[NSArray arrayWithObject:string]];
```

Snow Leopard Writing APIs

```
NSSound *sound = ...;

NSPasteboard *pboard = [NSPasteboard generalPasteboard];

[pboard clearContents];

[pboard writeObjects:[NSArray arrayWithObject:sound]];
```

NSPasteboardWriting

Three methods

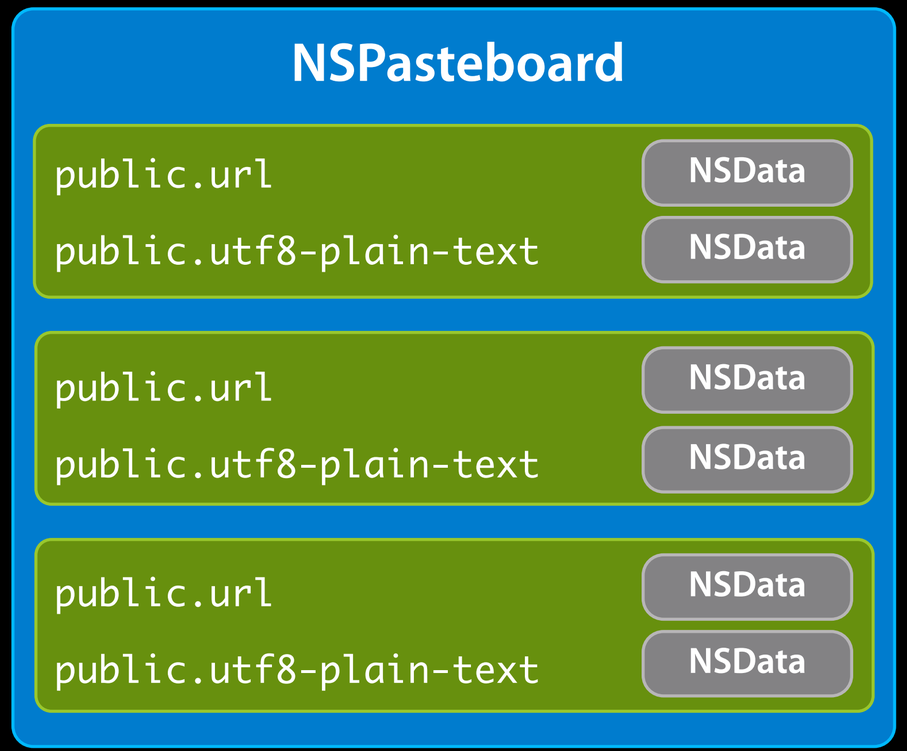
- -writableTypesForPasteboard:
- -pasteboardPropertyListForType:
- -writingOptionsForType:pasteboard:

Promising Data Using Writing Options

- By default
  - First type in provided array is written immediately
  - All other types are promised
- To customize, implement -writingOptionsForType:pasteboard: to return NSPasteboardWritingPromised or no option
- Objects that promise data are retained by the pasteboard

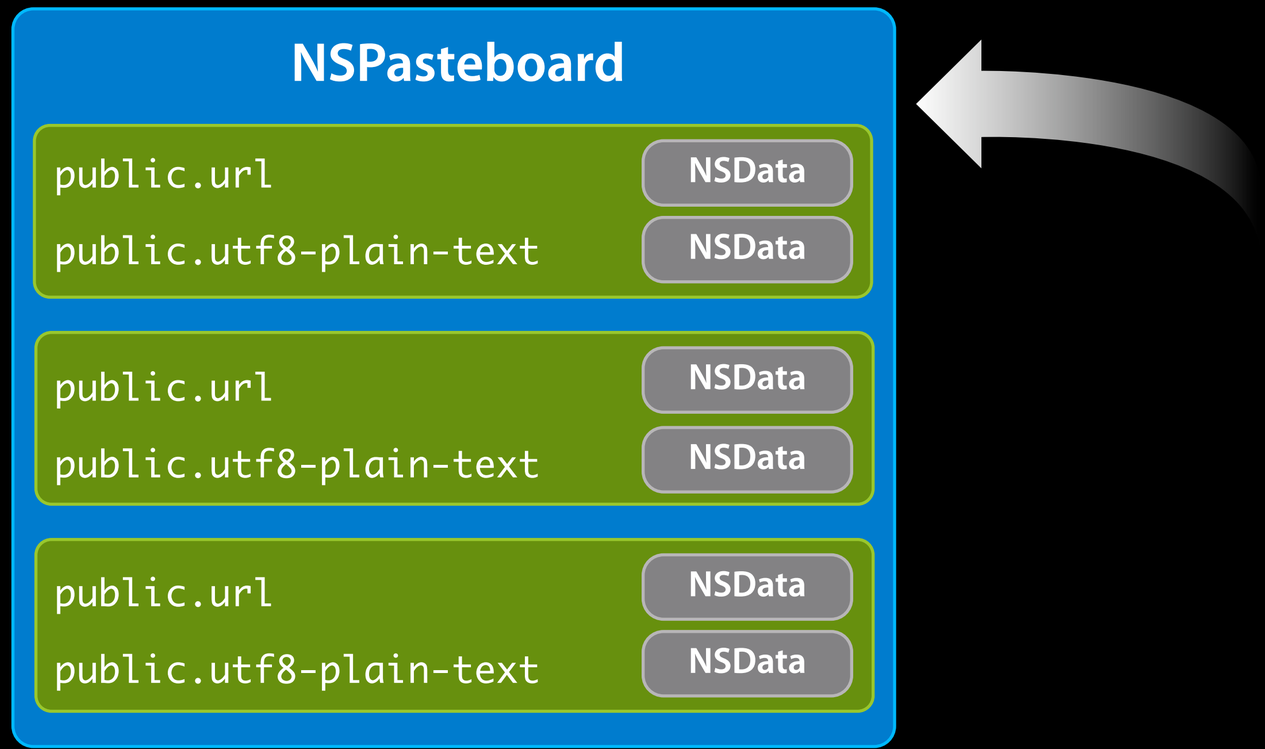
NSPasteboard

Reading from the pasteboard in Snow Leopard



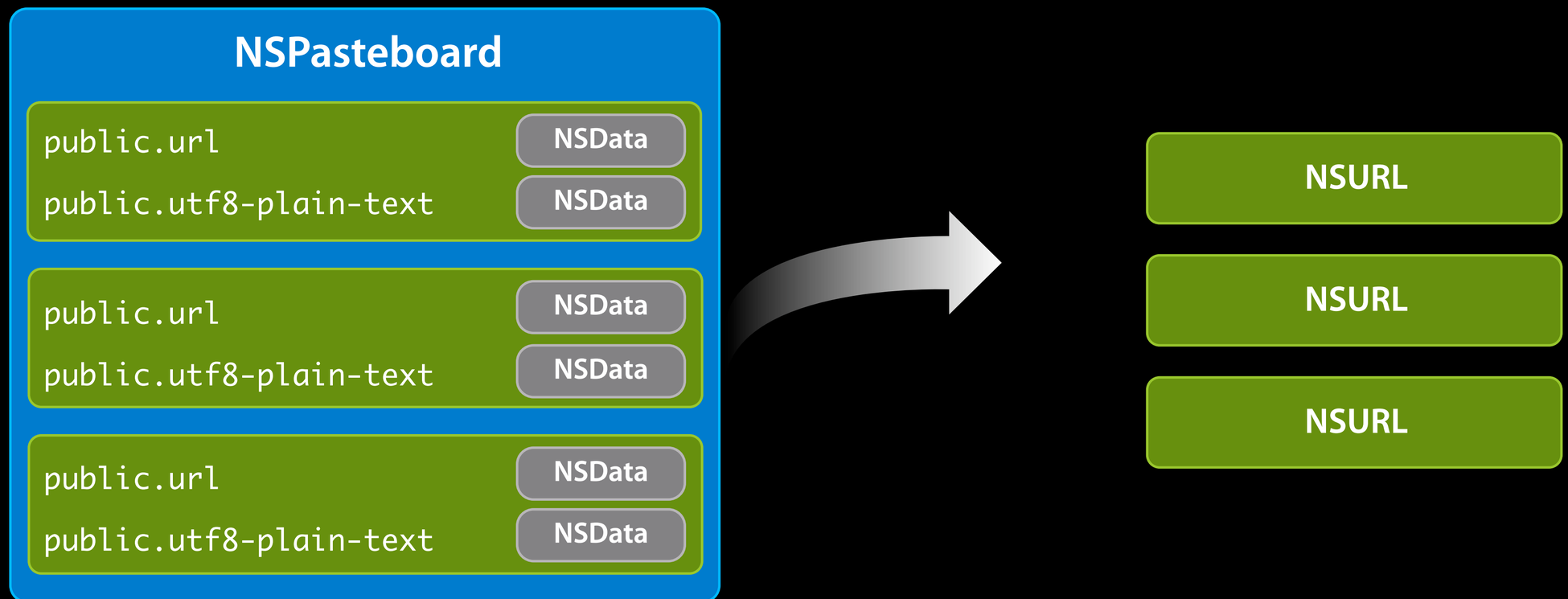
NSPasteboard

Reading from the pasteboard in Snow Leopard



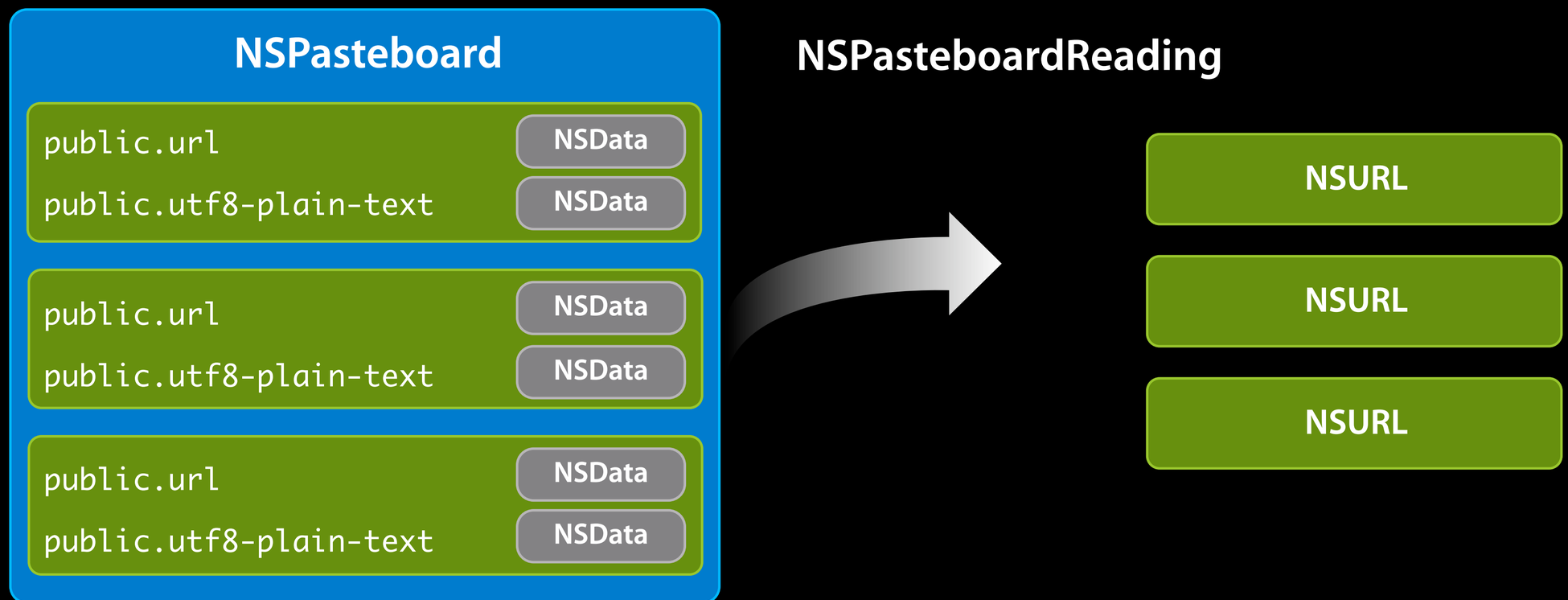
NSPasteboard

Reading from the pasteboard in Snow Leopard



NSPasteboard

Reading from the pasteboard in Snow Leopard



Snow Leopard Reading API

```
NSPasteboard *pboard = [NSPasteboard generalPasteboard];

NSArray *classes = [NSArray arrayWithObject:[NSURL class]];

NSArray *URLs =
            [pboard readObjectsForClasses:classes options:nil];
```

Snow Leopard Reading API

```
NSPasteboard *pboard = [NSPasteboard generalPasteboard];

NSArray *classes = [NSArray arrayWithObject:[NSURL class]];

NSArray *URLs =
            [pboard readObjectsForClasses:classes options:nil];
```

Snow Leopard Reading API

```
NSPasteboard *pboard = [NSPasteboard generalPasteboard];

NSArray *classes = [NSArray arrayWithObject:[NSURL class]];

NSArray *URLs =
            [pboard readObjectsForClasses:classes options:nil];
```

Snow Leopard Reading API

```
NSPasteboard *pboard = [NSPasteboard generalPasteboard];

NSArray *classes = [NSArray arrayWithObject:[NSURL class]];

NSArray *URLs =
            [pboard readObjectsForClasses:classes options:nil];
```

Snow Leopard Validation API

```
NSPasteboard *pboard = [NSPasteboard generalPasteboard];

NSArray *classes = [NSArray arrayWithObject:[NSURL class]];

BOOL canRead =
      [pboard canReadObjectForClasses:classes options:nil];
```

Snow Leopard Validation API

```
NSPasteboard *pboard = [NSPasteboard generalPasteboard];

NSArray *classes = [NSArray arrayWithObject:[NSURL class]];

BOOL canRead =
      [pboard canReadObjectForClasses:classes options:nil];
```

Snow Leopard Validation API

```
NSPasteboard *pboard = [NSPasteboard generalPasteboard];

NSArray *classes = [NSArray arrayWithObject:[NSURL class]];

BOOL canRead =
      [pboard canReadObjectForClasses:classes options:nil];
```

Snow Leopard Validation API

```
NSPasteboard *pboard = [NSPasteboard generalPasteboard];

NSArray *classes = [NSArray arrayWithObject:[NSURL class]];

BOOL canRead =
      [pboard canReadObjectForClasses:classes options:nil];
```

Read File URLs Only

Options Dictionary

`NSPasteboardURLReadingFileURLsOnlyKey` `[NSNumber numberWithBool:YES]`

NSPasteboard

- http://www.apple.com
- file:///Users/me/screenshot.png
- file:///Users/me/cat.jpg
- file:///Users/me/notes.rtf
- ftp://ftp.stanford.edu

NSURL

file:///Users/me/screenshot.png

NSURL

file:///Users/me/cat.jpg

NSURL

file:///Users/me/notes.rtf

Read Image File URLs Only

Options Dictionary

`NSPasteboardURLReadingFileURLsOnlyKey` `[NSNumber numberWithBool:YES]`

`NSPasteboardURLReadingContentsConformToTypesKey` `[NSImage imageTypes]`

NSPasteboard

- http://www.apple.com
- file:///Users/me/screenshot.png
- file:///Users/me/cat.jpg
- file:///Users/me/notes.rtf
- ftp://ftp.stanford.edu

NSURL
file:///Users/me/screenshot.png

NSURL
file:///Users/me/cat.jpg

NSPasteboardReading

- Three methods
  - +readableTypesForPasteboard:
  - -initWithPasteboardPropertyList:type:
  - +readingOptionsForType:pasteboard:
- Reading options

NSPasteboardReading

- Three methods
  - +readableTypesForPasteboard:
  - -initWithPasteboardPropertyList:type:
  - +readingOptionsForType:pasteboard:
- Reading options
  - NSPasteboardReadingAsData

NSPasteboardReading

- Three methods
  - +readableTypesForPasteboard:
  - -initWithPasteboardPropertyList:type:
  - +readingOptionsForType:pasteboard:
- Reading options
  - NSPasteboardReadingAsString

NSPasteboardReading

- Three methods
  - +readableTypesForPasteboard:
  - -initWithPasteboardPropertyList:type:
  - +readingOptionsForType:pasteboard:
- Reading options
  - NSPasteboardReadingAsPropertyList

NSPasteboardReading

- Three methods
  - +readableTypesForPasteboard:
  - -initWithPasteboardPropertyList:type:
  - +readingOptionsForType:pasteboard:
- Reading options
  - NSPasteboardReadingAsKeyedArchive

NSData

Initializes using -initWithCoder:
instead of -initWithPasteboardPropertyList:type:

More Fine-Grained Control

NSPasteboardItem

- New in Snow Leopard
- Interface similar to existing NSPasteboard single-item API
- Fine-grained control over exact types and data promised or provided
- Generic wrapper that can represent pasteboard data for any pasteboard item

NSPasteboardItem API

```
-setData:forType:
-setString:forType:
-setPropertyList:forType:
-setDataProvider:forTypes:

-types
-availableTypeFromArray:

-dataForType:
-stringForType:
-propertyListForType:
```

When to Use NSPasteboardItem

- When class to types mapping is not appropriate, for example:
  - Reading image data from the pasteboard to create a CGImage
  - Writing a custom plist type to the pasteboard
- When you receive a pasteboard as a delegate or subclasser
  - Use pasteboard items to inspect what is already on pasteboard
  - Use pasteboard items to provide or promise additional data

Using NSPasteboardItem

```
NSPasteboard *pboard = [NSPasteboard generalPasteboard];

NSPasteboardItem *item = [[NSPasteboardItem alloc] init];

[item setData:data forType:NSPasteboardTypeTIFF];

[item setDataProvider:self forTypes:
  [NSArray arrayWithObject:NSPasteboardTypePDF]];

NSArray *items = [NSArray arrayWithObject:item];

[pboard clearContents];

[pboard writeObjects:items];
```

Using NSPasteboardItem

```
NSPasteboard *pboard = [NSPasteboard generalPasteboard];

NSArray *items = [pboard pasteboardItems];

for (NSPasteboardItem *item in items) {

 if ([[item types] containsObject:NSPasteboardTypeTIFF]) {

    [item setDataProvider:self forTypes:
        [NSArray arrayWithTypes:NSPasteboardTypePDF]];

 }

}
```

NSPasteboardItem Notes

- Use as a temporary object, don’t hang onto it
- It is associated with only one pasteboard
- It is associated with only one change count

Transition to Uniform Type Identifiers (UTIs)

From pboard types to more modern UTIs

New NSPasteboard APIs Are UTI-Only

- Existing APIs continue to accept both pboard types and UTIs
- Moving to UTIs is straightforward
- Pboard types may be deprecated in a future release

Transitioning System Declared Types

- Common pasteboard types have new constants
- NSStringPboardType -> NSPasteboardTypeString
- For other types use UTI string constants defined in UTCoreTypes.h
- Convert uses of NSFilenamesPboardType to use new multiple item API
- See NSPasteboard.h for details about each type

Transitioning Custom Types

- Type strings must be in the form of a UTI
- You do not have to formally declare the UTI in your Info.plist file
  - Informal UTI strings are a good option for private pasteboard types
- If you have pboard types that third parties depend on
  - Formally declare the replacement UTI
  - Associate the new UTI with the old pboard type in Info.plist file
  - See AppKit 10.6 release notes for an example

More Types Are Now Available

- More types now possible on pasteboard
- Look for and accept more types
- For example, don’t just look for TIFF on the pasteboard, could now be any image type
- Use [NSImage imageTypes] for array of all supported UTIs
- Better yet, let the new NSPasteboard API take care of it for you

“Peter put a PICT upon the pasteboard.

Deprecated PICT’s a poor pasteboard type to pick.”

Cupertino Tongue Twister

PICT Is Deprecated!

- PICT is deprecated and unavailable in 64-bit
- Cocoa and Carbon will translate to and from PICT automatically
- No app should be explicitly reading or writing PICT on the pasteboard
- NSPICTPboardType is deprecated in Snow Leopard

Call to Action

- Move to UTIs
- Move from filenames to multiple file URLs
- Be prepared to receive multiple items
- Adopt the new API

Services

Peter Ammon
Cocoa Frameworks Engineer

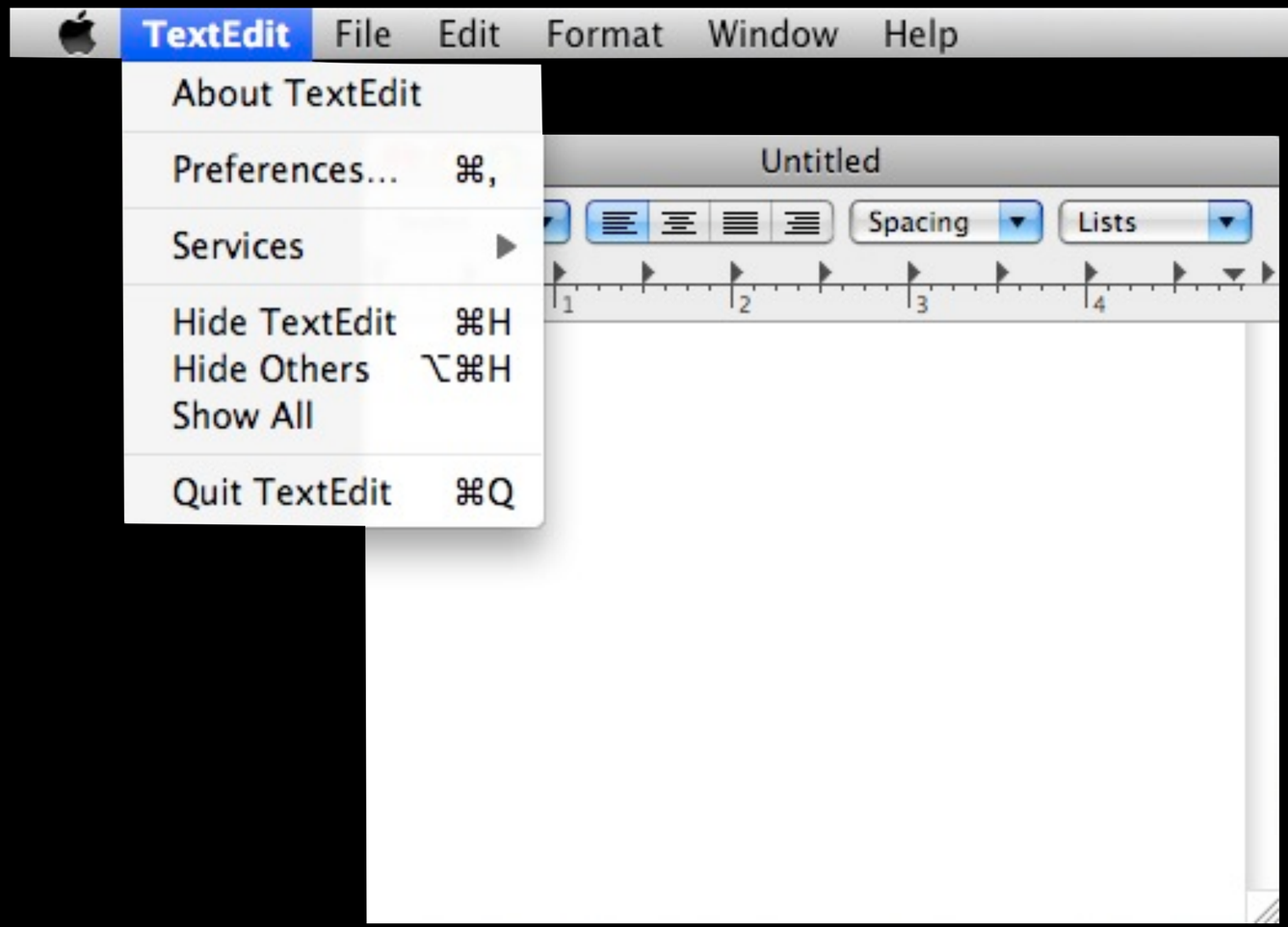
Services

- Services overview
- What's new in Snow Leopard
- Advertising a service
- Providing a service
- Consuming a service
- Debugging services
- Workflows as services

Services

- Services overview
- What's new in Snow Leopard
- Advertising a service
- Providing a service
- Consuming a service
- Debugging services
- Workflows as services

Services Overview



Services Overview

- Data is passed between the consumer and provider
- App functionality can be distributed and shared among other apps

Service Consumer

Service Provider

Any application can extend the functionality of many others.

Give example of Services, and stress how useful they are, while also talking about problems with Leopard Services menu

“Any application can extend the functionality of many others.”

NeXTSTEP 3.3 Developer Documentation, 1995

Services

- Services overview
- What's new in Snow Leopard
- Advertising a service
- Providing a service
- Consuming a service
- Debugging services
- Workflows as services

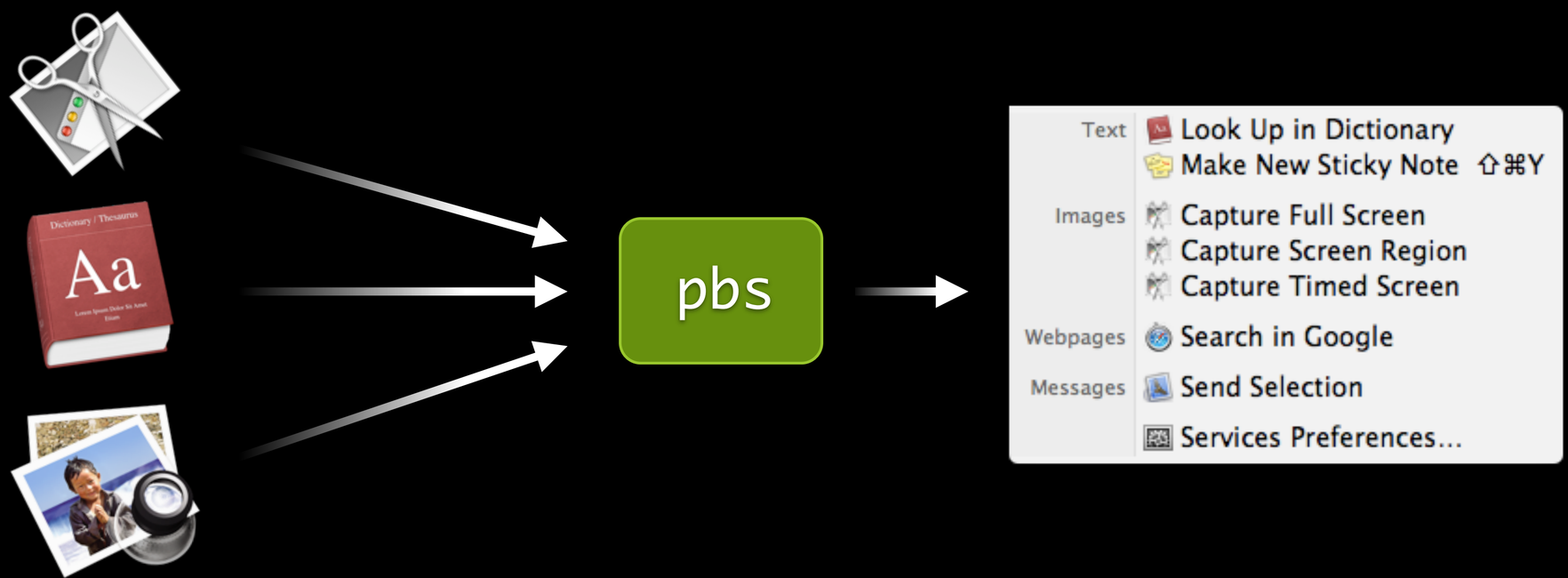
Demo

Services

- Services overview
- What's new in Snow Leopard
- Advertising a service
- Providing a service
- Consuming a service
- Debugging services
- Workflows as services

How the Services Menu Is Built

- You declare Services in your app's Info.plist
- pbs scans apps and caches the Services declarations
- AppKit builds the Services menu



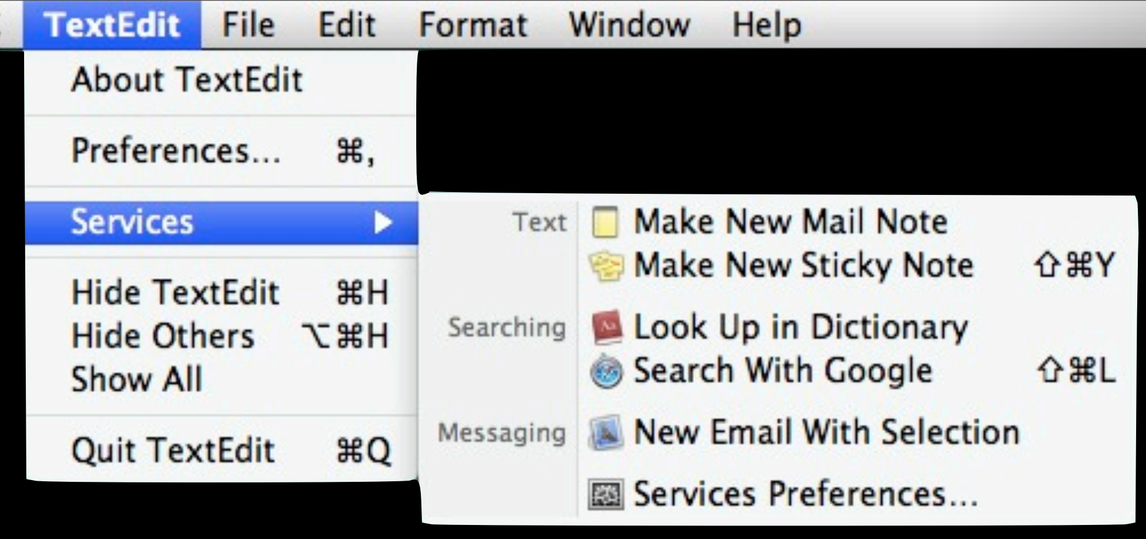
How the Services Menu Is Built

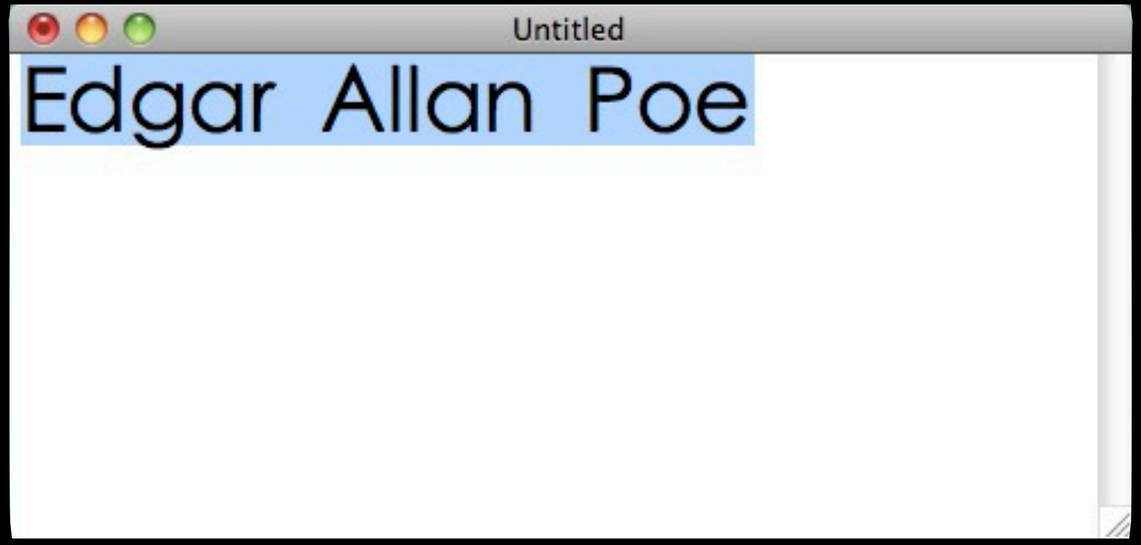
```
<key>NSServices</key>
<array>
    <dict>
        <key>NSMenuItem</key>
        <dict>
            <key>default</key>
            <string>Open Selected File</string>
        </dict>
        <key>NSMessage</key>
        <string>openFile</string>
        <key>NSSendTypes</key>
        <array>
            <string>public.plain-text</string>
        </array>
```

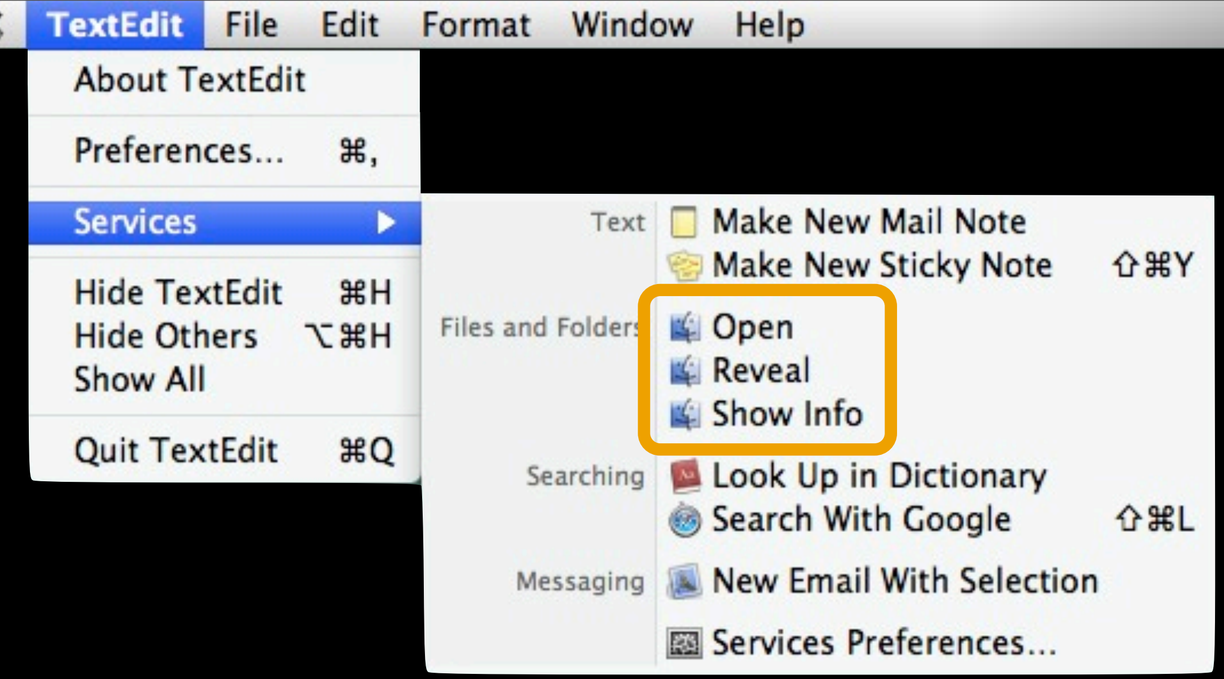
How the Services Menu Is Built

```
<key>NSServices</key>
<array>
    <dict>
        <key>NSMenuItem</key>
        <dict>
            <key>default</key>
            <string>Open Selected File</string>
        </dict>
        <key>NSMessage</key>
        <string>openFile</string>
        <key>NSSendTypes</key>
        <array>
            <string>public.plain-text</string>
        </array>
        <key>NSRequiredContext</key>
```

Determines whether your Service is enabled by default (even if empty)


TextEdit
File
Edit
Format
Window
Help
About TextEdit
Preferences... ⌘,
Services
Hide TextEdit ⌘H
Hide Others ⌥⌘H
Show All
Quit TextEdit ⌘Q
Text
Make New Mail Note
Make New Sticky Note ⇧⌘Y
Searching
Look Up in Dictionary
Search With Google ⇧⌘L
Messaging
New Email With Selection
Services Preferences...

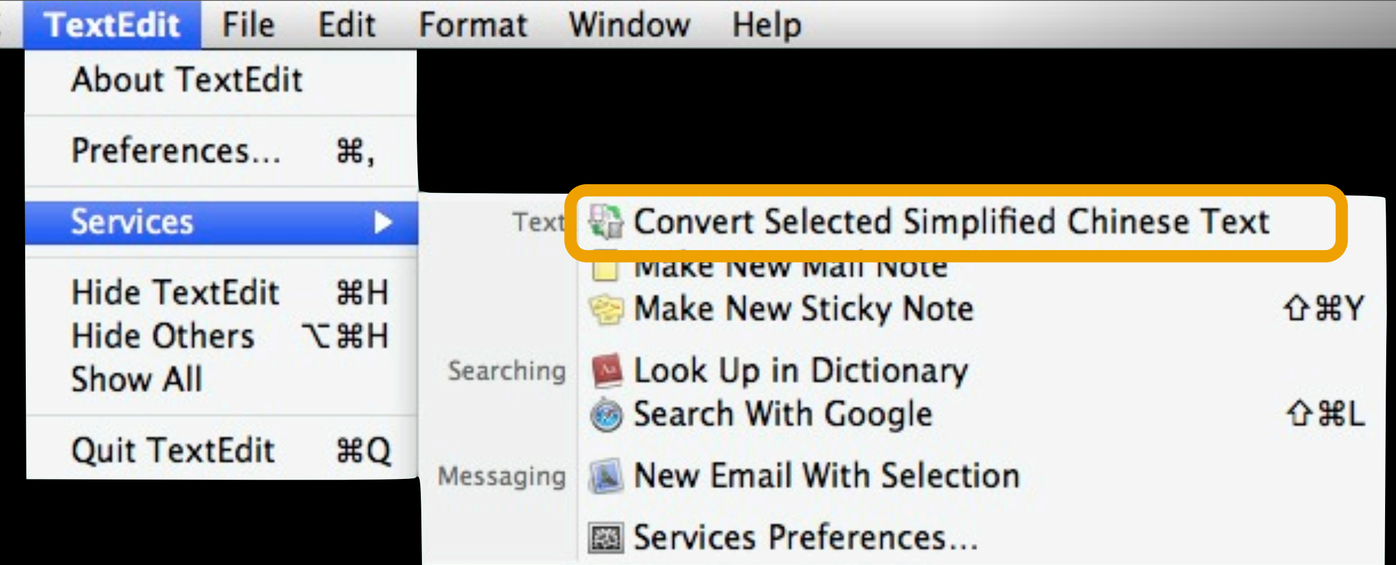

Untitled
Edgar Allan Poe



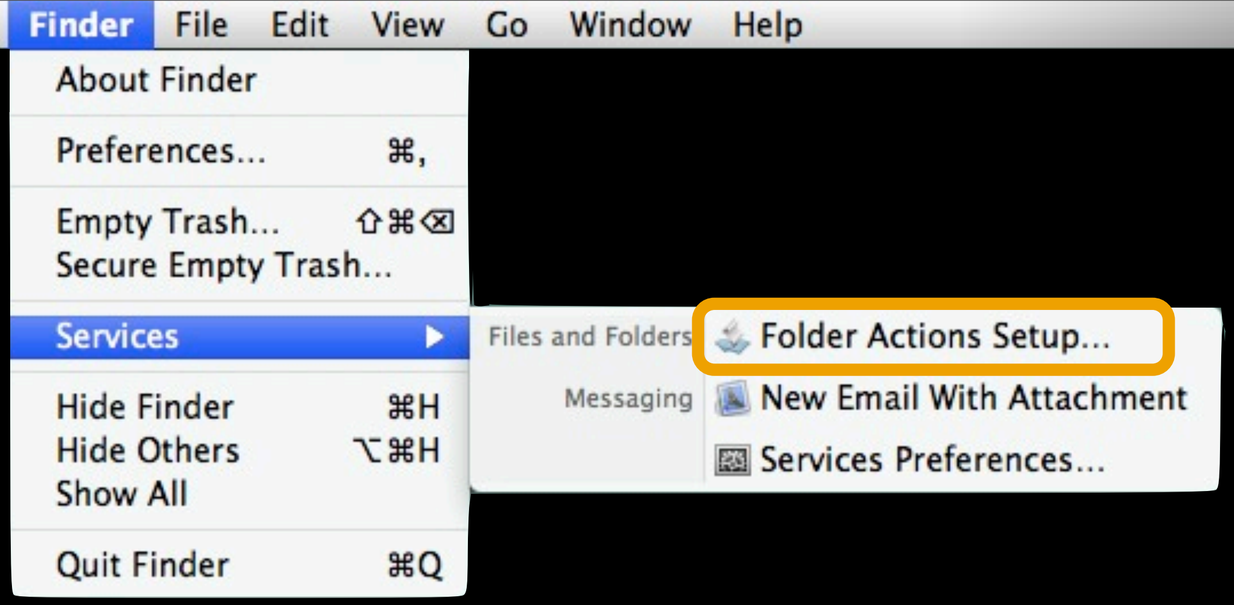

```
<key>NSRequiredContext</key>
<dict>
  <key>NSTextContent</key>
  <string>FilePath</string>
</dict>
```

Supported NSTextContent types

- URL
- Address
- Date
- Email
- FilePath




```
<key>NSRequiredContext</key>
<dict>
  <key>NSTextScript</key>
  <string>Hans</string>
</dict>
```



```
<key>NSSendFileTypes</key>
<array>
  <string>public.folder</string>
</dict>
```

- NSSendFileTypes describes file types
- Array of universal type identifiers (UTIs)
- Your Service gets file URLs on the pasteboard

More Contextuality

```
<key>NSRequiredContext</key>
<dict>
  <key>NSApplicationIdentifier</key>
  <string>com.apple.Xcode</string>
  <key>NSWordLimit</key>
  <int>1</int>
</dict>
```

Demo

Services

- Services overview
- What's new in Snow Leopard
- Advertising a service
- Providing a service
- Consuming a service
- Debugging services
- Workflows as services

Congratulations!

The user clicked your Service. Now what?

After the Click

Service Consumer

Service Provider

```
  NSSendTypes: public.plain-text
NSReturnTypes: public.tiff
```

After the Click

Bridezilla

NSPasteboard

Service Consumer

Service Provider

```
  NSSendTypes: public.plain-text
NSReturnTypes: public.tiff
```

After the Click

NSPasteboard

Bridezilla

Service Consumer

Service Provider

```
  NSSendTypes: public.plain-text
NSReturnTypes: public.tiff
```

After the Click

NSPasteboard

Bridezilla

Service Consumer

Service Provider

```
  NSSendTypes: public.plain-text
NSReturnTypes: public.tiff
```

After the Click

NSPasteboard

Service Consumer

Service Provider

```
  NSSendTypes: public.plain-text
NSReturnTypes: public.tiff
```

After the Click

- Your app will be launched if necessary
- Your app should register its “service provider”

```
[NSApp setServiceProvider:provider];
```

- The provider receives a message depending on the Service

```
- (void)openSelection:(NSPasteboard *)pboard
             userData:(NSString *)data
                error:(NSString **)error
```

```
<key>NSMessage</key>
<string>openSelection</string>
<key>NSUserData</key>
<string>data</string>
```

Demo

After the Click

- If your Service has no return type, the process is asynchronous
- If your Service has a return type, the calling app waits
- User can cancel in Snow Leopard with Escape or ⌘-.

Services

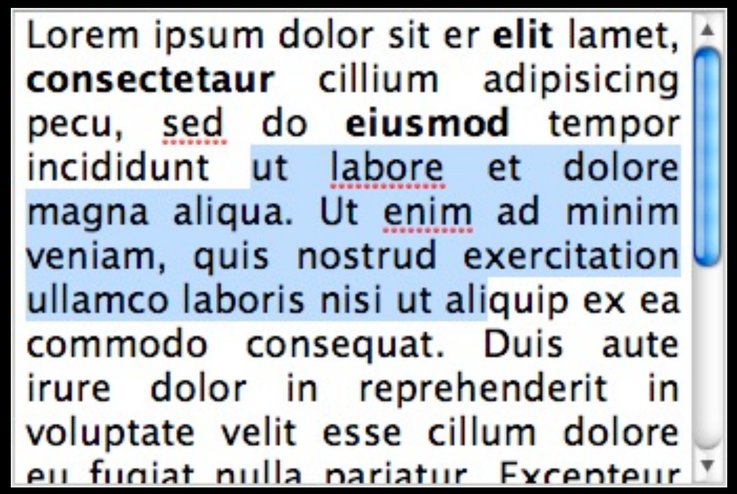
- Services overview
- What's new in Snow Leopard
- Advertising a service
- Providing a service
- Consuming a service
- Debugging services
- Workflows as services

Consuming a Service

- Services are more useful than ever
- Users will start to expect them to work wherever they make sense
- Your views can implement an API to support Services

Standard Controls

NSTextView



WebView

If you can use either of these, your work is done.

Custom Controls

1. Register the types you support
2. Indicate when you support those types
3. Write to and/or read from NSPasteboard

Demo

Services

- Services overview
- What's new in Snow Leopard
- Advertising a service
- Providing a service
- Consuming a service
- Debugging services
- Workflows as services

Debugging Services

- The pbs tool is your window into Services registration
- Run it with no arguments to immediately scan for Services changes

```
/System/Library/CoreServices/pbs
```

- It will also typecheck your Info.plist
- Run it with -dump_pboard to list all registered Services

```
/System/Library/CoreServices/pbs -dump_pboard
```

- pbs is for debugging only!

Debugging Services

- The NSDebugServices user default is your window into Services at runtime

```
TextEdit -NSDebugServices com.yourcompany.yourapp
```

- It will output information about why a Service does or does not appear

Demo

Services

- Services overview
- What's new in Snow Leopard
- Advertising a service
- Providing a service
- Consuming a service
- Debugging services
- Workflows as services

Demo

Workflows as Services

- Users can turn workflows into Services
- Learn about exposing your app as Automator actions
  - “Using Automator and Services to Integrate with Mac OS X”

Using Automator and Services to Integrate with Mac OS X	North Beach Friday 2:00PM

Summing Up

- NSPasteboard has new APIs in Snow Leopard
  - Common operations require much less code
  - Shift focus from types to classes
  - Shift from pboard types to UTIs
- Services are rejuvenated in Snow Leopard
- Some ways to integrate
  - Providing contextual Services from your app
  - Allowing Services to be consumed within your app
  - Exposing your app as Automator actions

More Information

Matt Drance
Application Frameworks Evangelist
mdrance@apple.com

Documentation

Services Implementation Guide
Pasteboard Programming Guide for Mac OS X v10.6 and Later

Related Sessions

Using Automator and Services to Integrate with Mac OS X	North Beach Friday 2:00PM

Labs

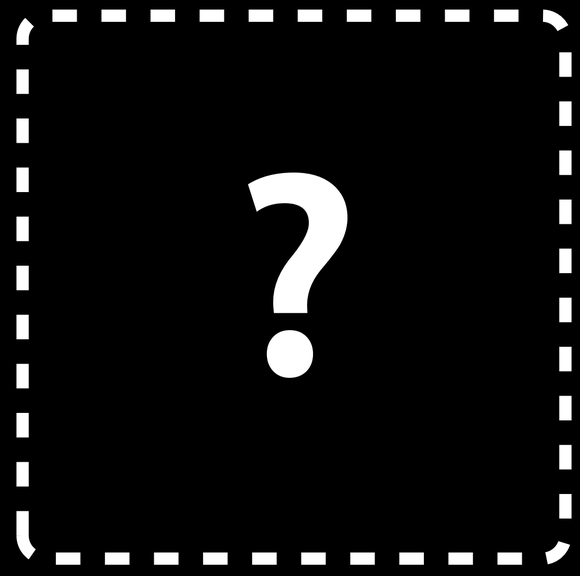
Cocoa Open Lab	OS X Lab C Thursday 2:00PM

Song

James Dempsey and the Breakpoints

NEWSLETTER
Check out the great Pasteboard and Services session at WWDC 2009.
It is sure to be a barn burn of a session as James Dempsey and Peter Ammon present what is new in pasteboard and services for Snow Leopard.

ps cax | grep "Finder"


?

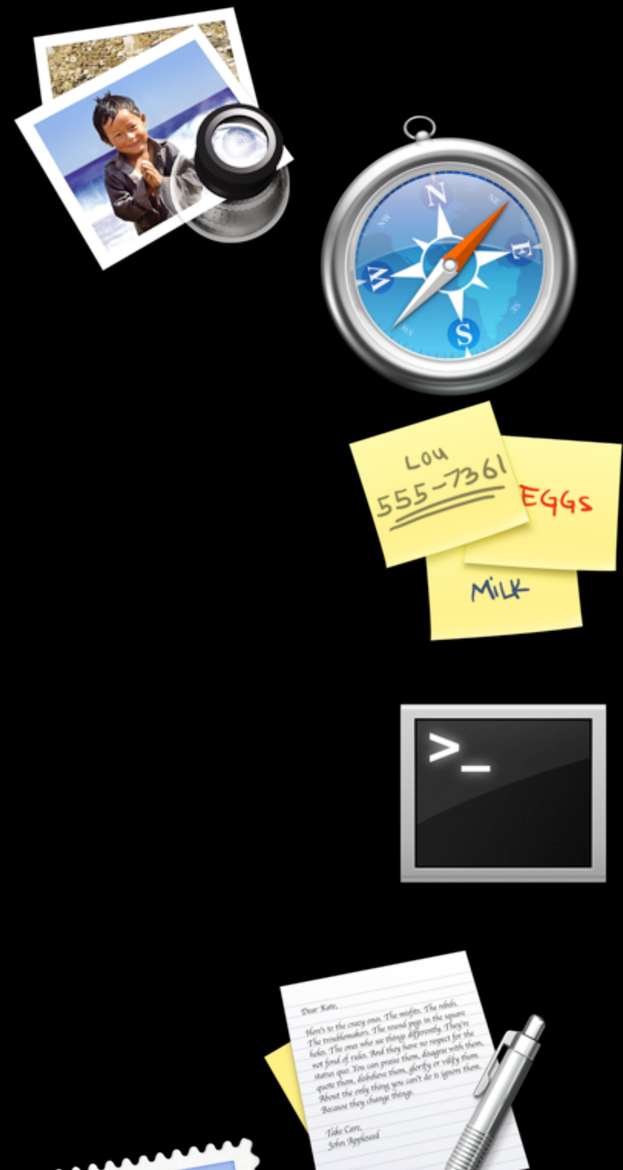
Lou
555-7361
EGGS
MILK

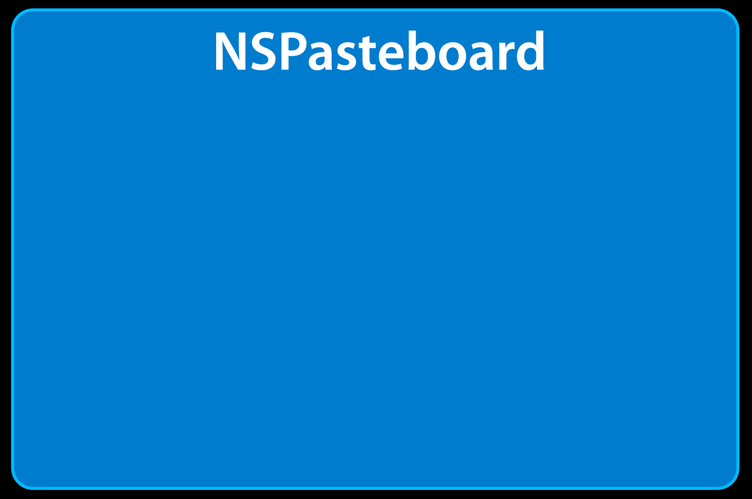
>_

Dear Katie,
Here's to the crazy ones. The misfits. The rebels.
The troublemakers. The round pegs in the square
holes. The ones who see things differently. They're
not fond of rules. And they have no respect for the
status quo. You can praise them, disagree with them,
quote them, disbelieve them, glorify or vilify them.
About the only thing you can't do is ignore them.
Because they change things.
Take Care,
John Appleseed

NSPasteboard

NSPasteboard




NSPasteboard

NSPasteboard
AppKit

NSPasteboard

NSPasteboard
AppKit

MyCustomClass

NSPasteboard

cordially invites

NSPasteboard

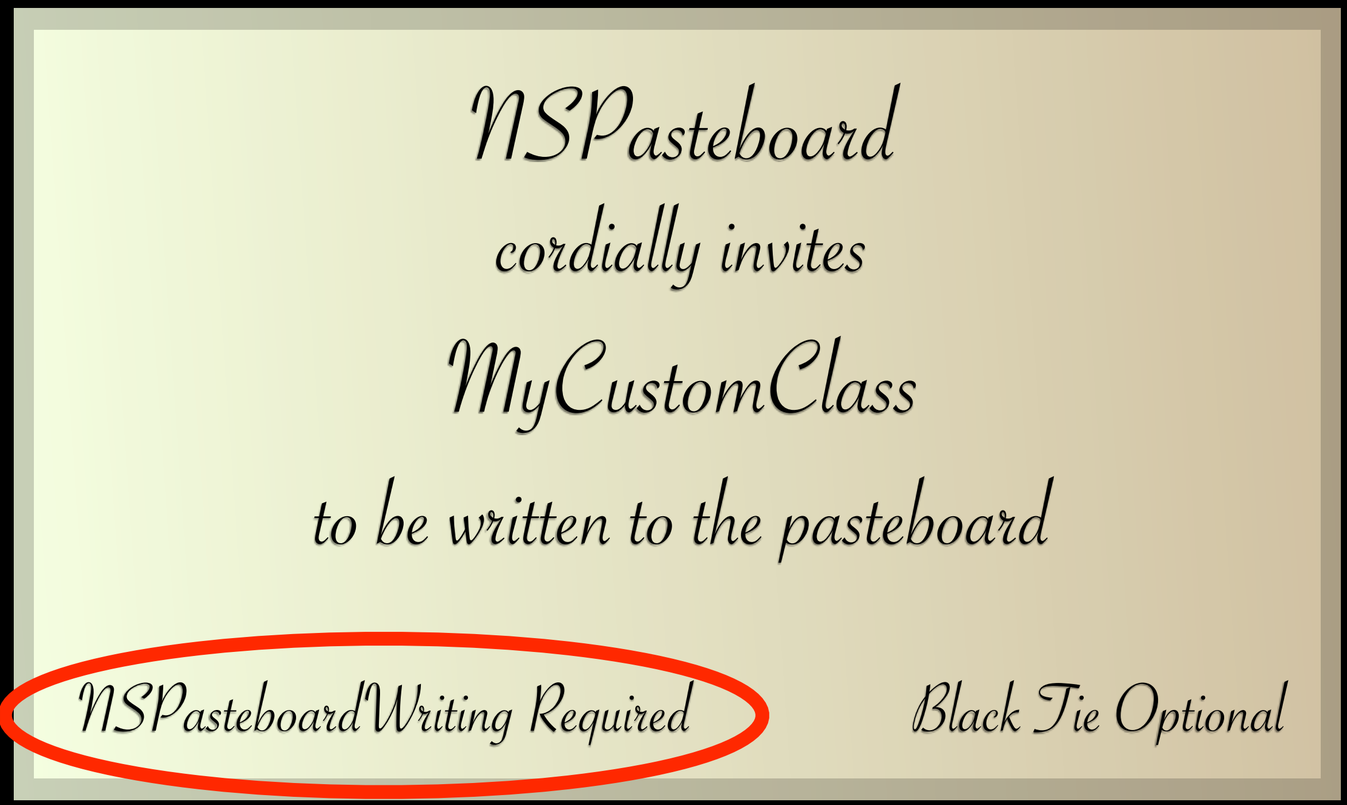
cordially invites

MyCustomClass

to be written to the pasteboard

NSPasteboardWriting Required

Black Tie Optional




```
-writableTypesForPasteboard:
-pasteboardPropertyListForType:
-writingOptionsForType:pasteboard:
```

NSColor

NSAttributedString

NSPasteboard

NSSound

NSString

NSImage

NSURL

NSColor

NSAttributedString

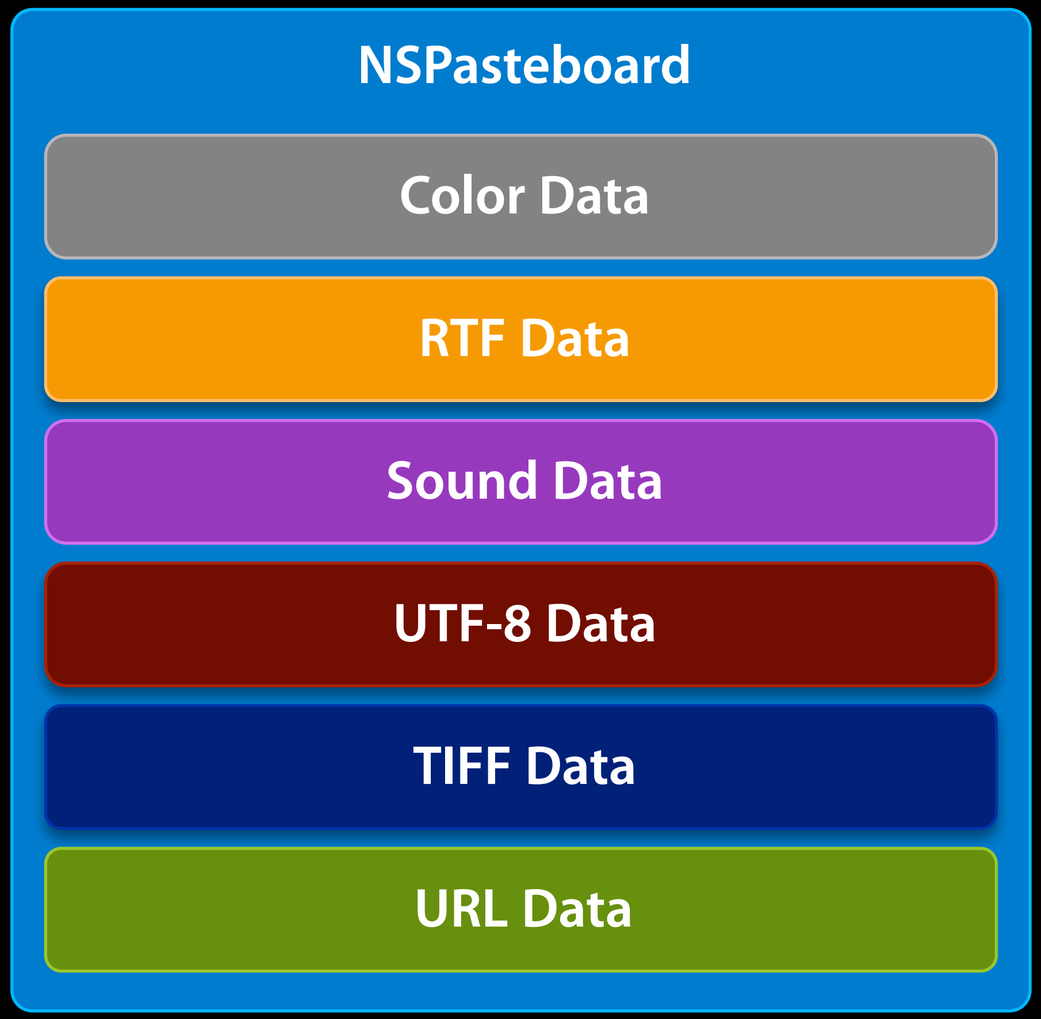
NSSound

NSString

NSImage

NSURL

NSPasteboard


NSPasteboard
Color Data
RTF Data
Sound Data
UTF-8 Data
TIFF Data
URL Data

Write ‘em

```
NSArray *URLs = ...;

NSPasteboard *pboard = [NSPasteboard generalPasteboard];

[pboard clearContents];

[pboard writeObjects:URLs];
```

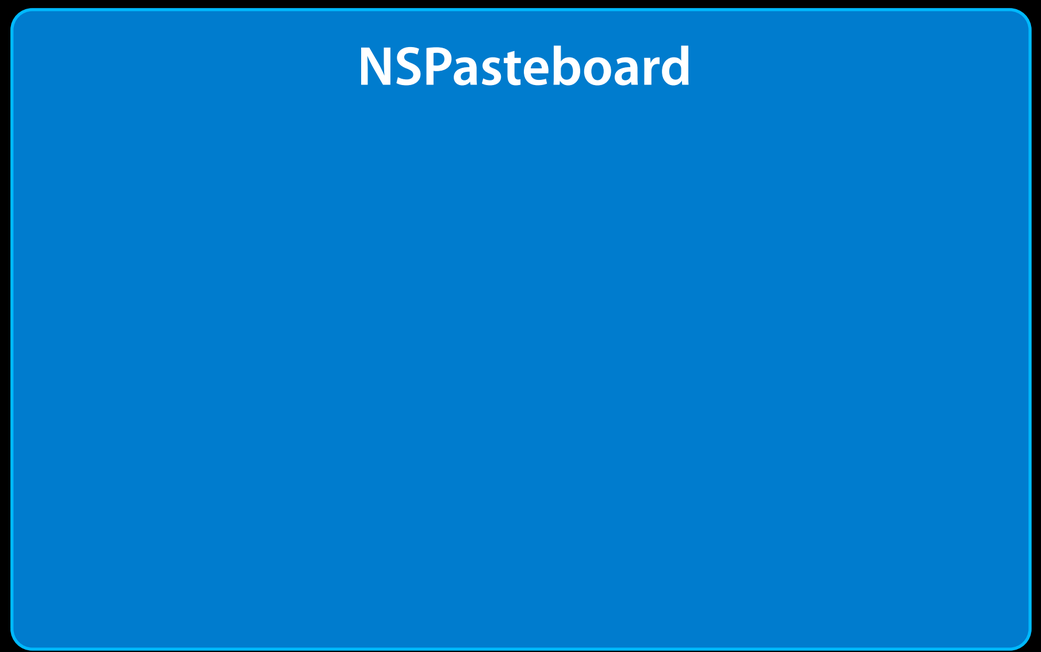
```
NSArray *URLs = ...;

NSPasteboard *pboard = [NSPasteboard generalPasteboard];

[pboard clearContents];

[pboard writeObjects:URLs];
```

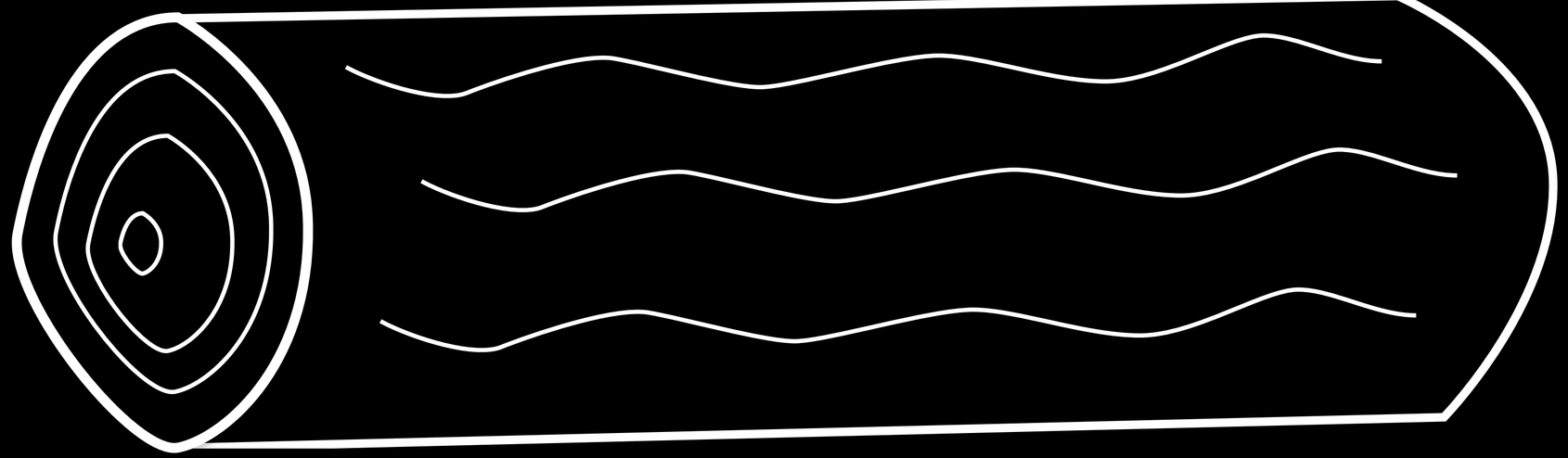
```
NSArray *URLs = ...;

NSPasteboard *pboard = [NSPasteboard generalPasteboard];

[pboard clearContents];

[pboard writeObjects:URLs];
```

1

2

3


NSPasteboard



A Proverbial Log

NO

PASTEBOARD

MISUSE

AT

ANYTIME

Don’t Eat The Paste

Thoughts of an NSPasteboard

Developer Done Me Wrong

It seems like some things never go right. It seems like some things

Archives

- June 2009
- May 2009
- April 2009
- March 2009
- February 2009
- January 2009
- December 2008
- November 2008
- October 2008
- September 2008
- August 2008
- July 2008

STOP

TIFF

PNG

JPEG

NSPasteboard

NSData

NSPasteboard

NSData

Thanks
Pboard!

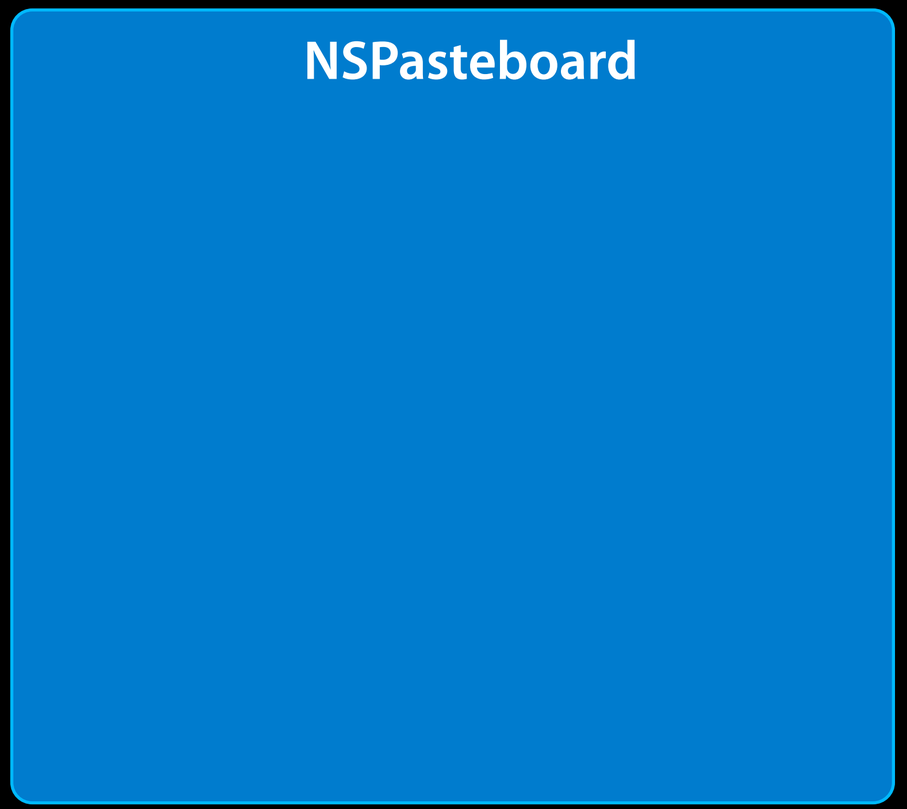
Congrats on
the new API!
iTunes
Music Card
$50

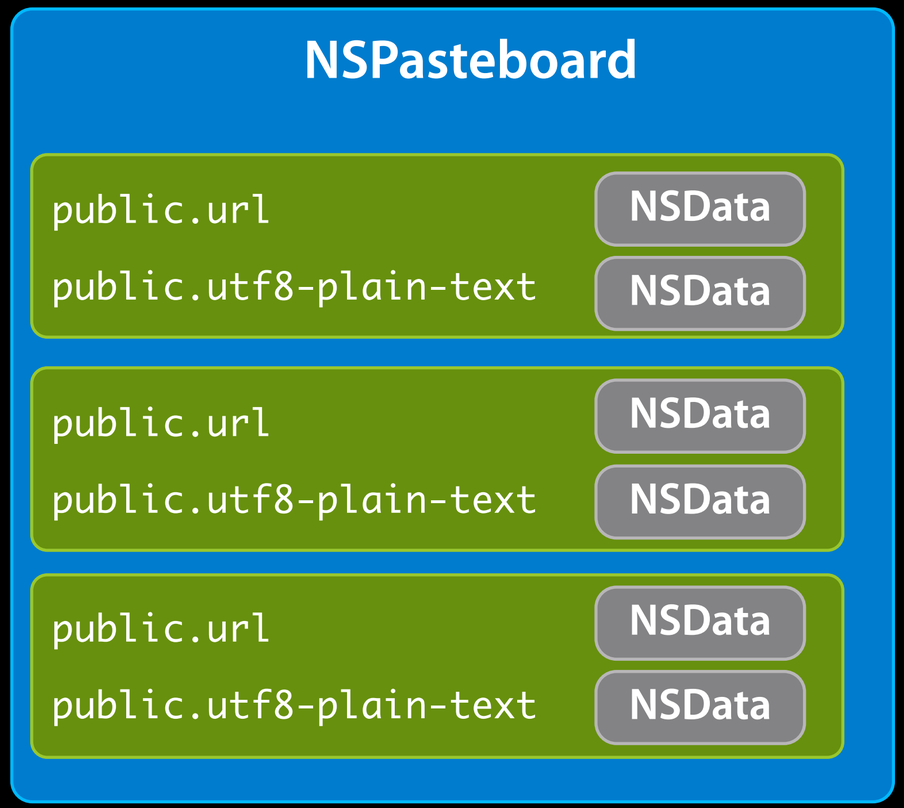
NSPasteboardReading

```
+readableTypesForPasteboard:
-initWithPasteboardPropertyList:type:
+readingOptionsForType:pasteboard:
```

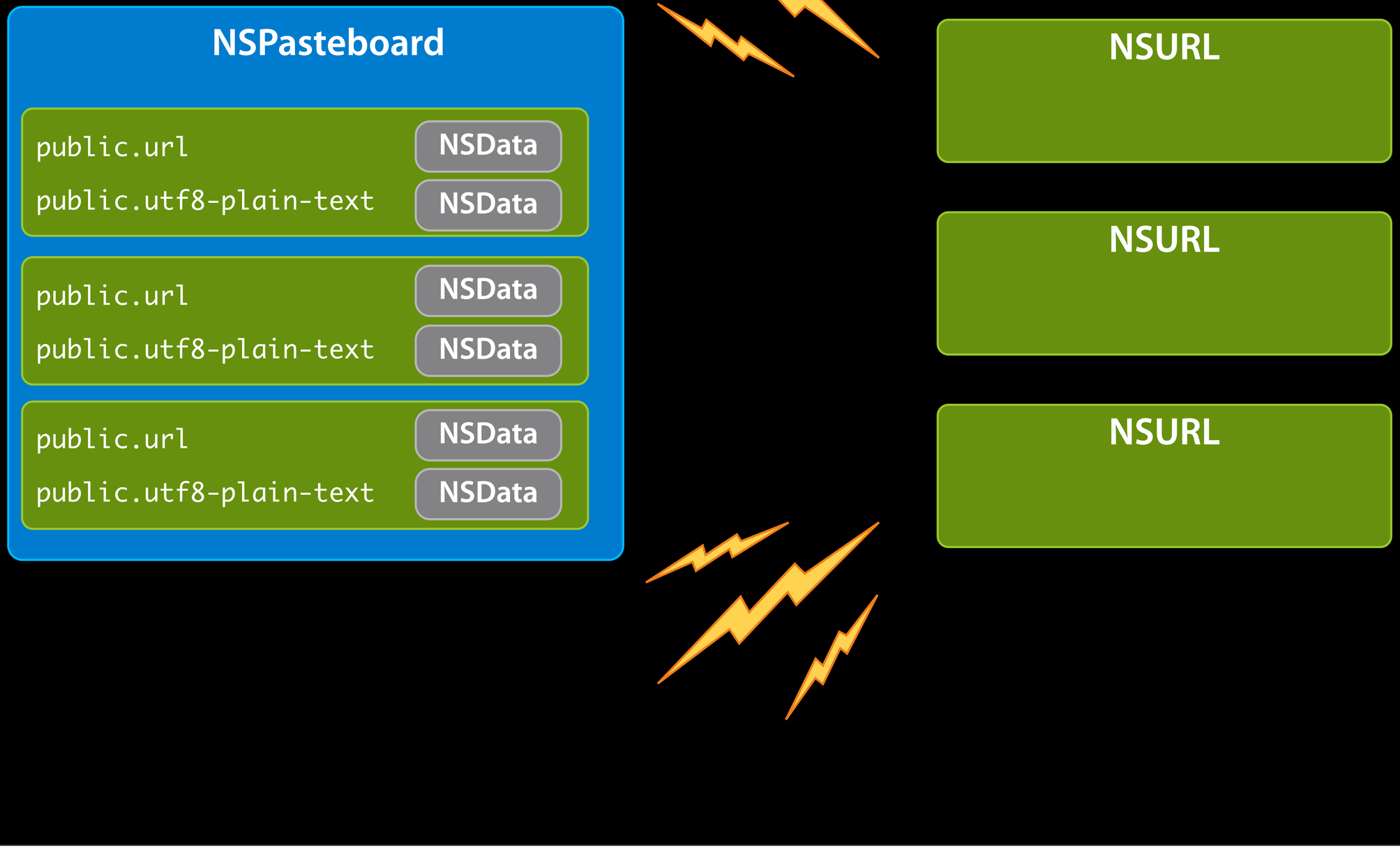
NSPasteboard

NSURL


NSPasteboard


NSPasteboard
public.url
NSData
public.utf8-plain-text
NSData
public.url
NSData
public.utf8-plain-text
NSData
public.url
NSData
public.utf8-plain-text
NSData

Do you have any NSURLs?


NSPasteboard
public.url
NSData
public.utf8-plain-text
NSData
public.url
NSData
public.utf8-plain-text
NSData
public.url
NSData
public.utf8-plain-text
NSData
NSURL
NSURL
NSURL

Read Baby, Read!

```
NSPasteboard *pboard = [NSPasteboard generalPasteboard];

NSArray *classes = [NSArray arrayWithObject:[NSURL class]];

NSArray *URLs =
        [pboard readObjectsForClasses:classes options:nil];
```

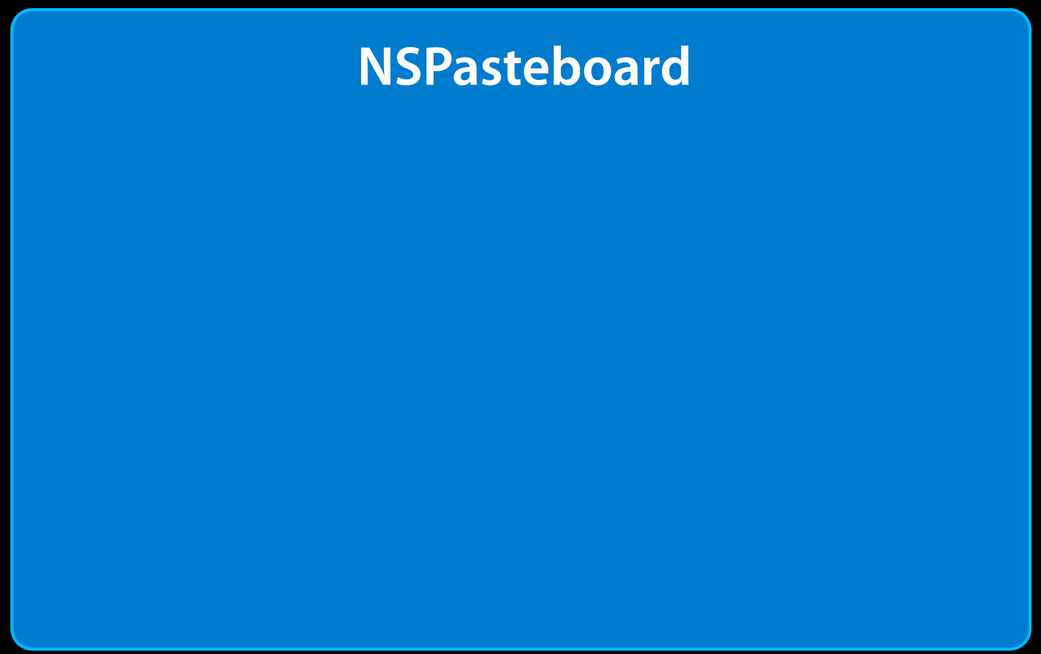
```
NSPasteboard *pboard = [NSPasteboard generalPasteboard];

NSArray *classes = [NSArray arrayWithObject:[NSURL class]];

NSArray *URLs =
        [pboard readObjectsForClasses:classes options:nil];
```

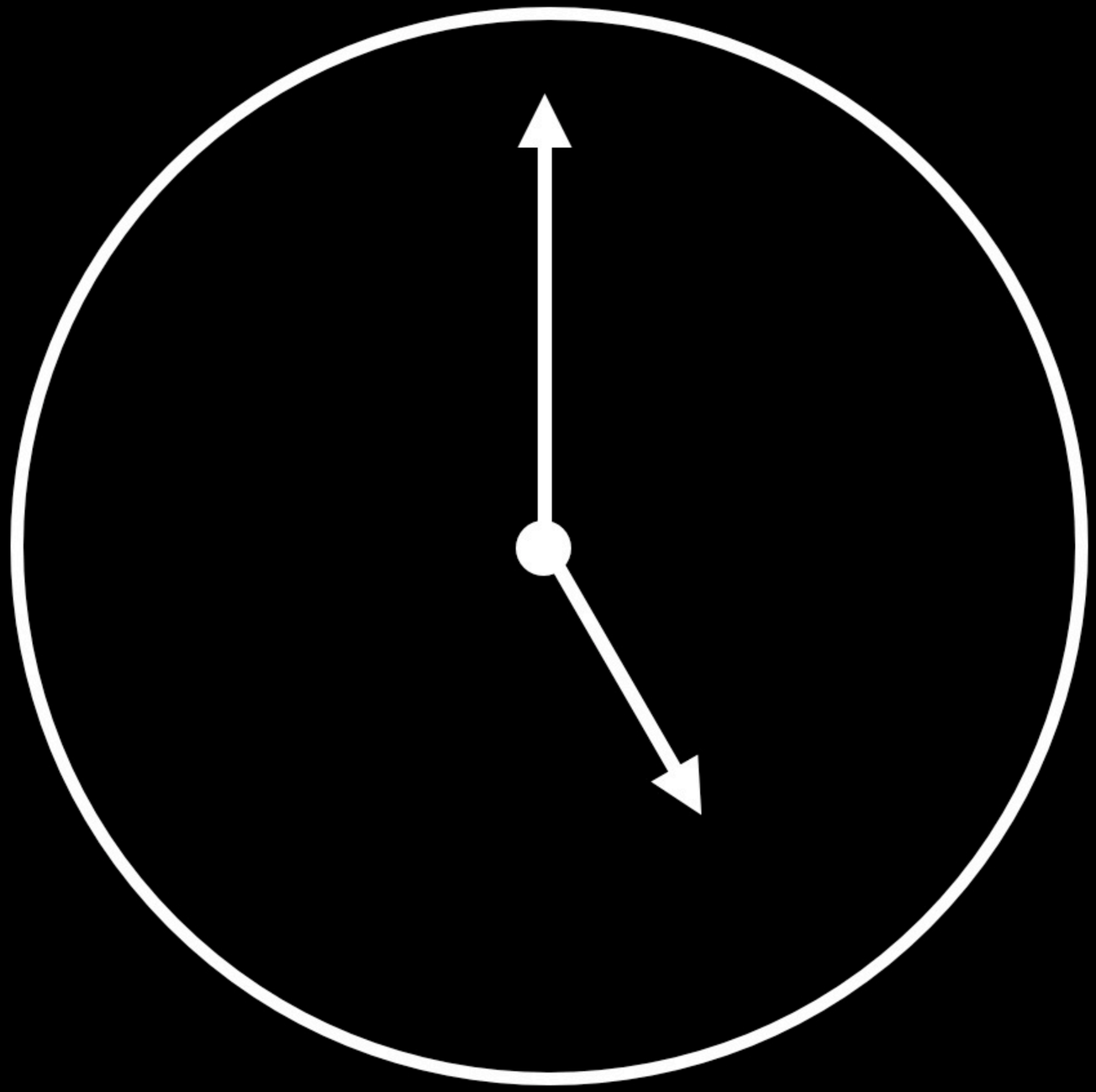
```
NSPasteboard *pboard = [NSPasteboard generalPasteboard];

NSArray *classes = [NSArray arrayWithObject:[NSURL class]];

NSArray *URLs =
        [pboard readObjectsForClasses:classes options:nil];
```

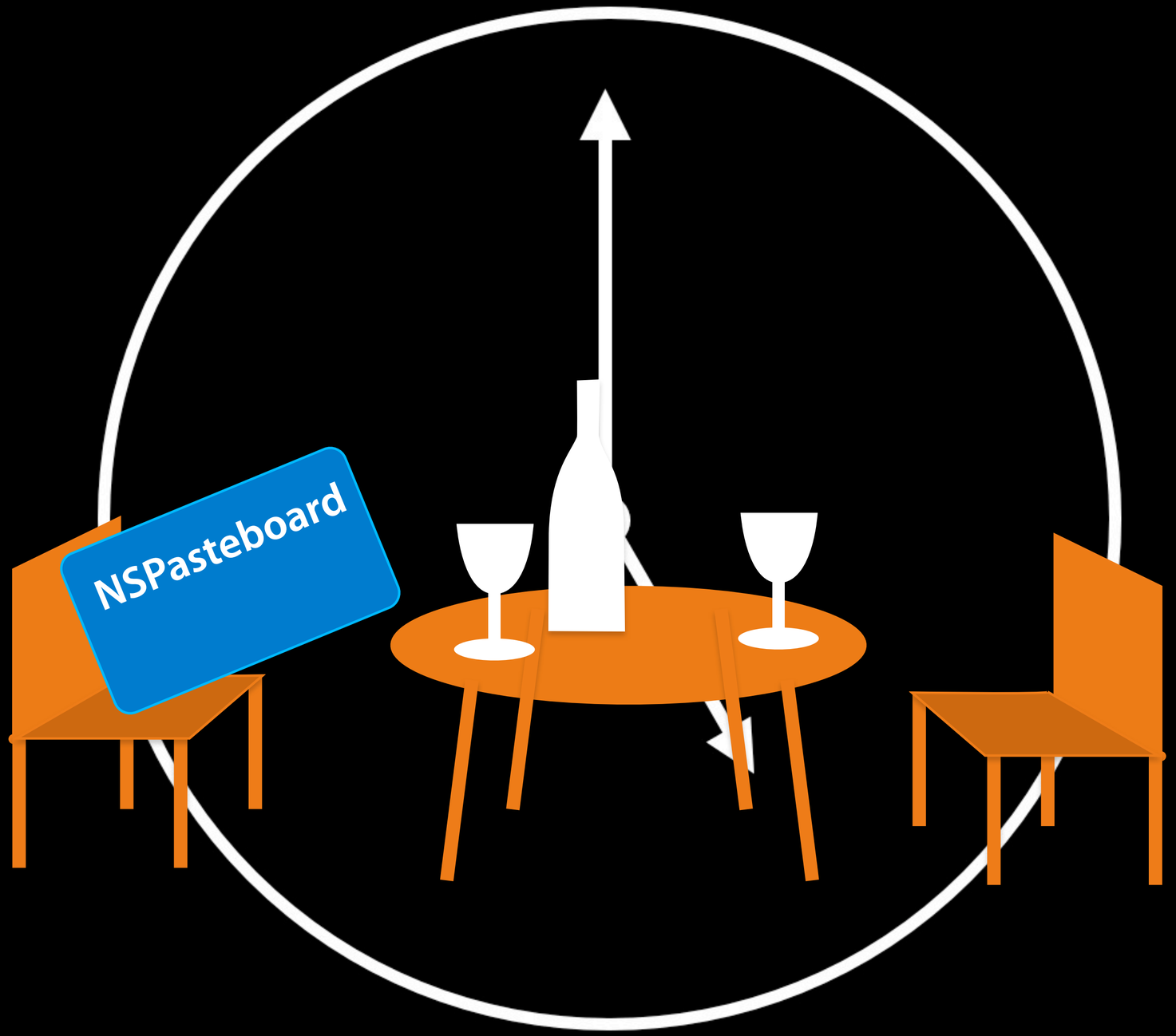
1

2

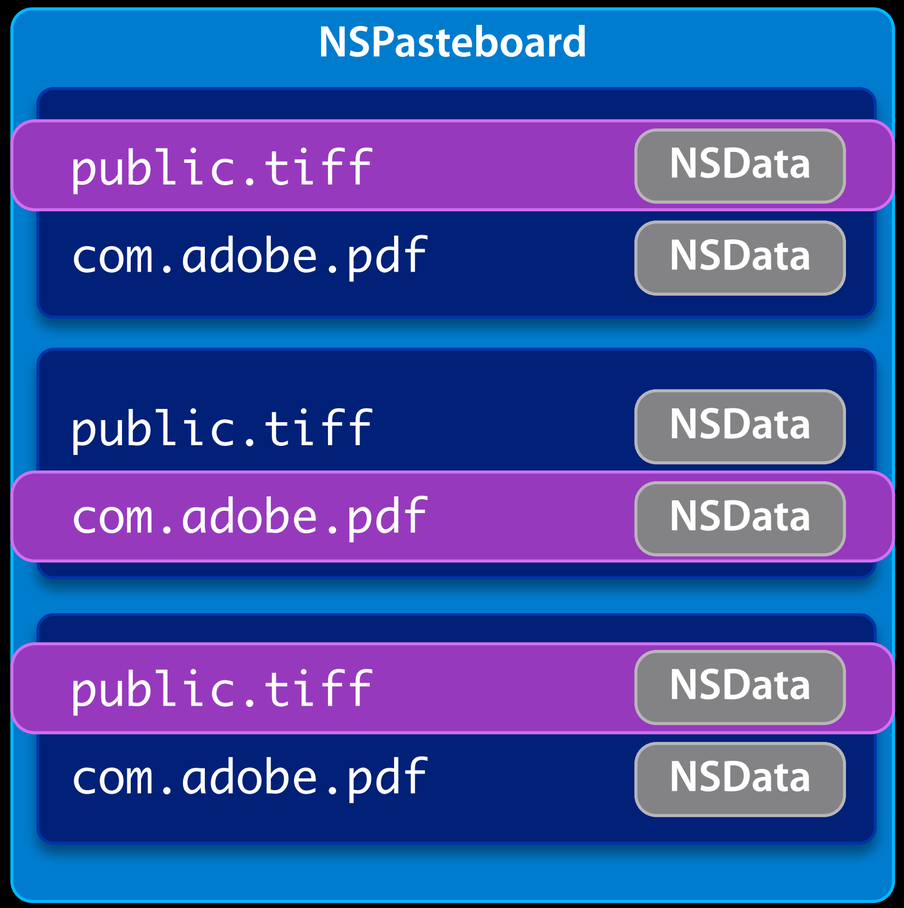
3


NSPasteboard




NSPasteboard

```
2 * read != write
```



I need that…

…that…

…and that.

NSPasteboardItem

public.tiff	NSData
com.adobe.pdf	NSData

NSPasteboardItem

public.tiff	NSData
com.adobe.pdf	NSData

NSPasteboardItem

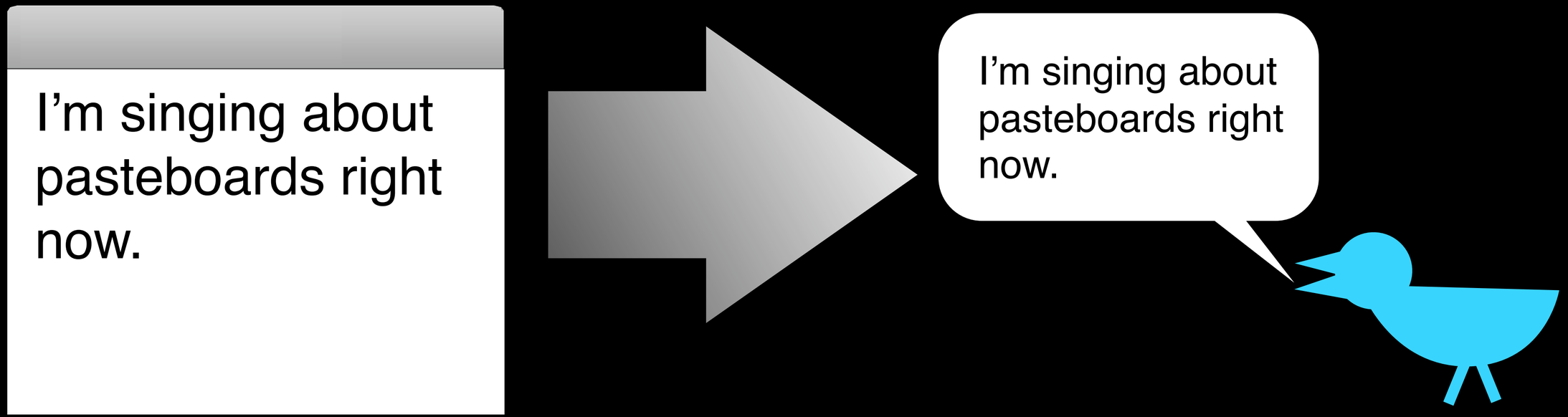
public.tiff	NSData
com.adobe.pdf	NSData

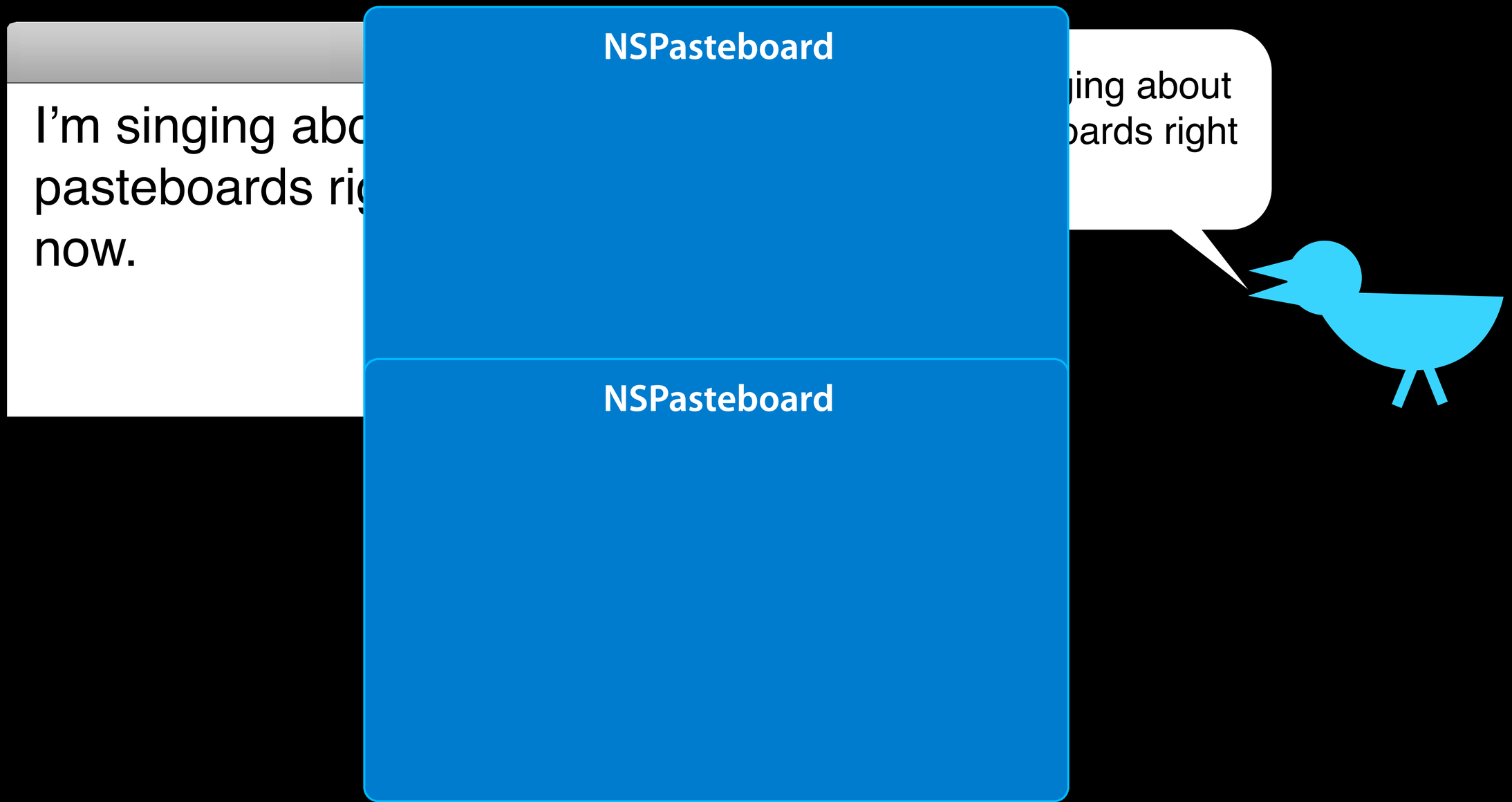
HELLO

MY NAME IS

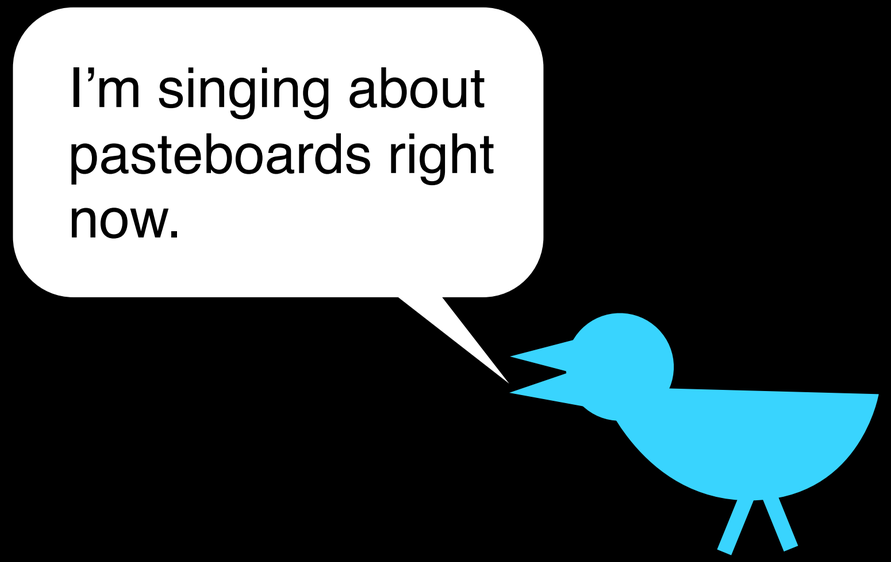
Control Freak

Service


I’m singing about pasteboards right now.
I’m singing about pasteboards right now.


I’m singing abo
pasteboards ri
now.
NSPasteboard
NSPasteboard
ging about
boards right

I’m singing about pasteboards right now.



I’m singing about pasteboards right now.

NSPasteboard

I’m singing about pasteboards right now.

NSPasteboard

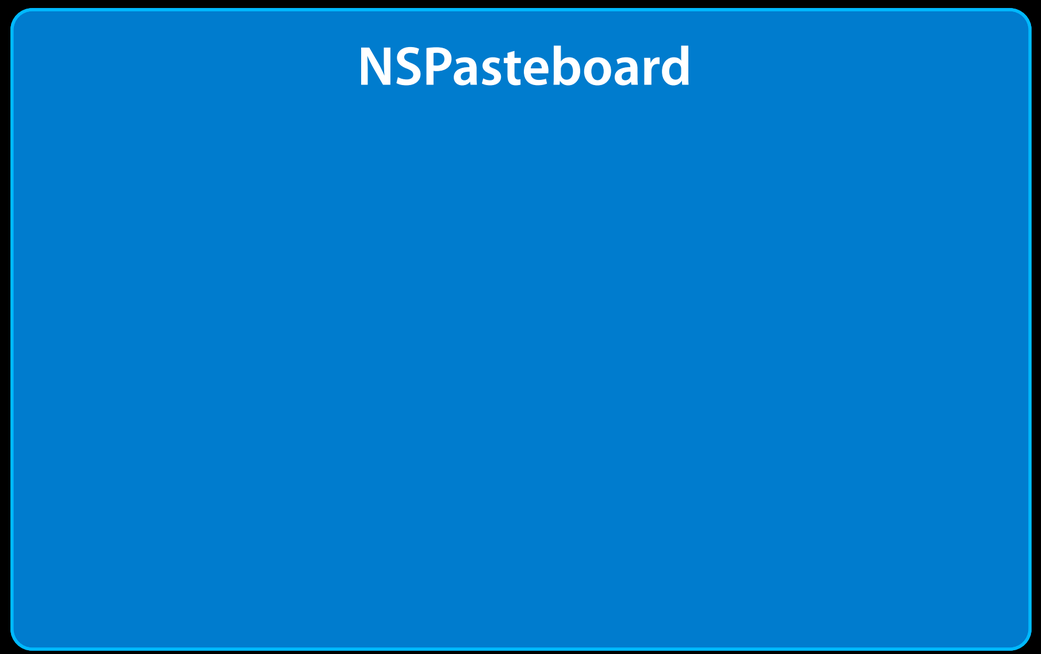
I’m singing about pasteboards right now.

NSPasteboard

I’m singing abo
pasteboards ri
now.

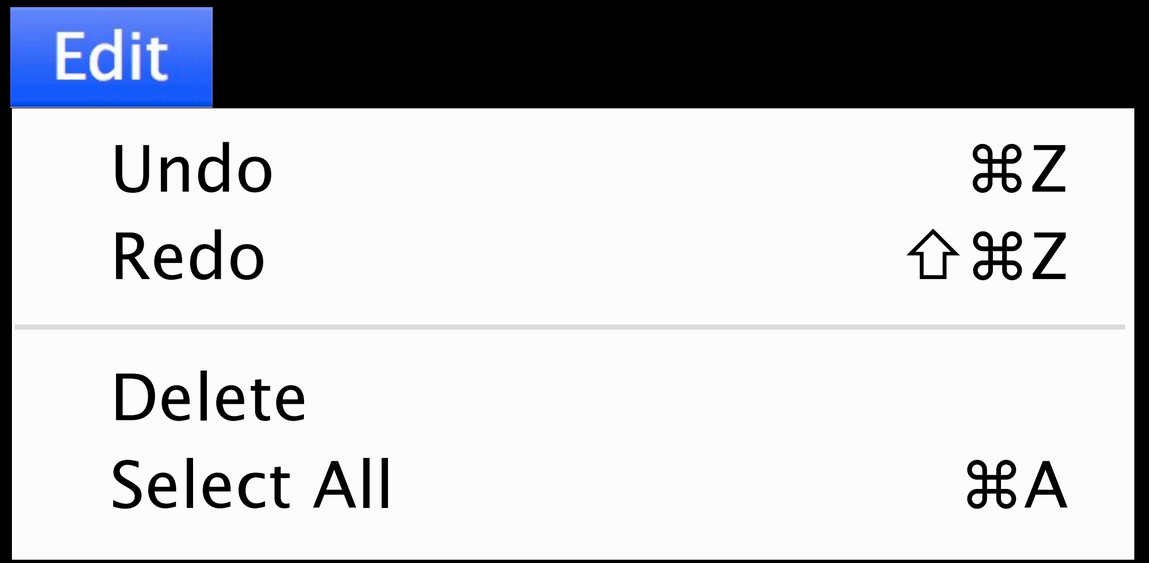
I’m singing about pasteboards right now.

n singing about
asteboards right
now.


NSPasteboard

Edit

Undo	⌘Z
Redo	⇧⌘Z
Cut	⌘X
Copy	⌘C
Paste	⌘V
Delete	
Select All	⌘A


Edit
Undo ⌘Z
Redo ⇧⌘Z
Delete
Select All ⌘A

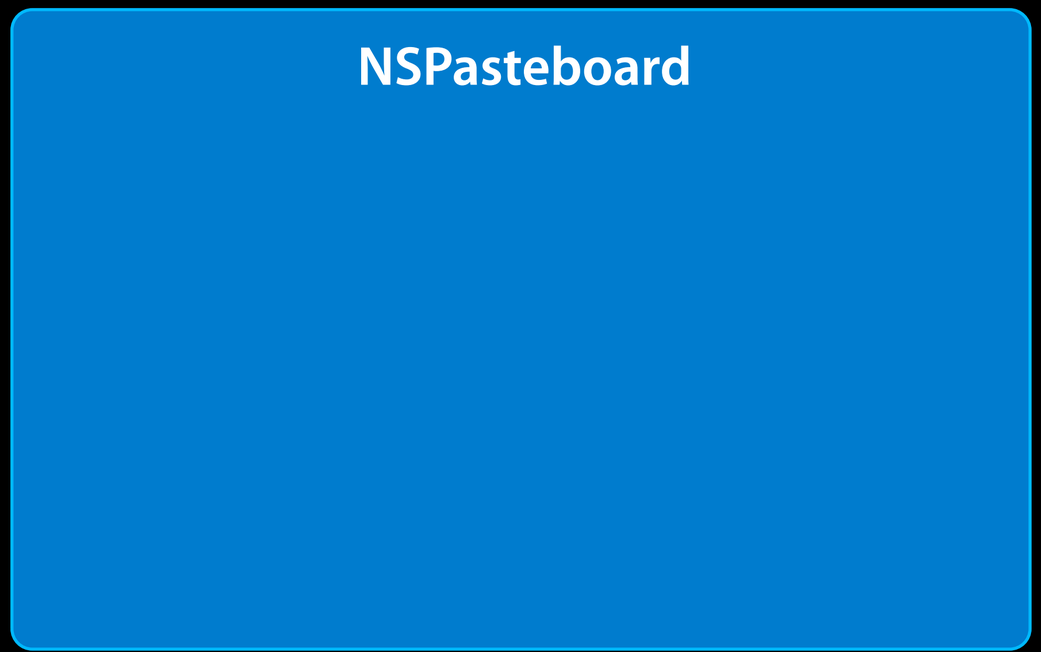
NSPasteboard

Where'd the pasteboard go?

NSPasteboard

We want the pasteboard!

We want the pasteboard!


NSPasteboard

NSPasteboard

+

You

=

Awesome App

Q&A

WWDC Session Survey

You can rate your sessions on the WWDC Attendee Site.

Survey forms are located on each session description page at:

developer.apple.com/wwdc/attendee

WWDC09